Group Members:

Khwaja Shahzaib (30700)

Abdullah Taufiq (29121)

PROJECT REPORT

BLOCK CHAIN

Email Storage and Retrieval System with Metamask Integration

Project Overview

The aim of this project is to create a decentralized email storage and retrieval system using blockchain technology. The application will be built using HTML/CSS for the frontend, JavaScript for the logic, and Metamask for blockchain integration. The system will allow user to securely store and retrieve his email address on the blockchain, ensuring transparency and immutability.

Implementation

- First Install Visual studio in your system for frontend using Html , css and Javascript.
- After that install node js and then Install Lite-server to run code from CLI.
- Download Metamask extension in your system and setup it by creating password and wallet.
- Select Currency and add faucets in that in my case I select Mumbai Polygon.
- After that go to their VS code and create their front end and validation and add web3 integration script in their head section and remaining script for contract address and ABI add in main Body.
- After completing their email registration form go to remix IDE and create their contract insert or fetch their inserted email for verification.
- After creating their contract compile it and select **Injected Provider-MetaMask** to alert MetaMask for transaction and then deploy it.
- Copy its address and contract ABI file and paste it in web3 script address and ABI section for connectivity.
- Then type **lite-server** in vs code terminal to run code after that display pop up in your browser.

Project Features

1. **Metamask Integration:**

- Utilize the Metamask wallet for user authentication and transaction management.
- Enable users to connect their Metamask wallets to the application securely.
- MetaMask Extension:

  https://chromewebstore.google.com/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn

2. **Email Storage Form:**
   - Create a user-friendly form using HTML to collect email address from user.
   - Implement validation checks on the client side to ensure the correctness of the provided email address.
3. **Blockchain Smart Contract:**
   - Develop a smart contract using Solidity for the Ethereum blockchain.
   - Include functions to securely store and retrieve email address on the blockchain.
   - Source Link is: https://remix.ethereum.org/
4. **Web3 Integration:**
   - Use Web3.js to interact with the Ethereum blockchain from the frontend.
   - Enable seamless communication between the frontend and the smart contract for data storage and retrieval.
   - Source Link is: https://web3js.readthedocs.io/en/v1.10.0/web3-eth-accounts.html

```
<script src="https://cdn.jsdelivr.net/npm/web3@1.3.0/dist/web3.min.js"></script>
```

- Add this above script on html head section

```
        const web3 = new Web3(window.ethereum);
    const contractAddress = '0x6a387a581f94D02dc51E665AF5D85dbE2767A33D'; //
Replace with your deployed smart contract address
    const contractABI =[//ABI Code];
```

- Add this script in html body to run the solidity code by adding contract address and ABI file.

5. **User Interface:**
   - Design a clean and intuitive user interface using HTML/CSS to enhance user experience.
   - Display transaction status and relevant information to users during the email storage and retrieval process.
6. **Security Measures:**
   - Implement security measures to protect user data and ensure the integrity of transactions.
   - Utilize encryption techniques to safeguard sensitive information.

Technology Stack

1. **Frontend:**

   a. HTML5
   b. CSS
   c. Javascript

2. **Blockchain Integration:**

   a. Ethereum

   b. Solidity

3. **Blockchain Interaction:**

   a. Web3.js

4. **Wallet Integration:**

   a. Metamask

5. **Development Environment:**

   a. Visual Studio Code

6. **MetaMask Wallet Currency:**

   a. Mumbai Polygon

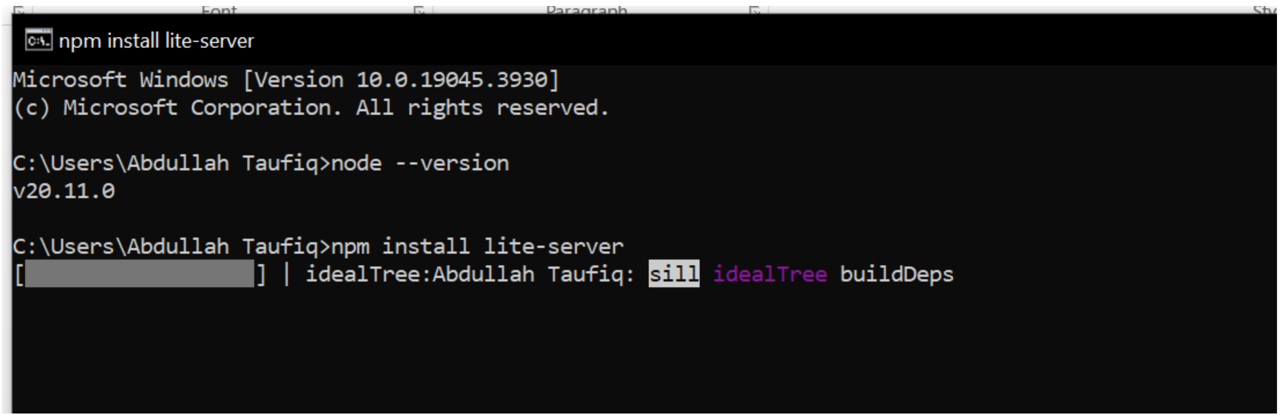
   b. https://mumbaifaucet.com/

7. **Node js & Lite server**

   a. Node Js first install in system and then for integration go to CLI for for further integration and also set their environment by providing their path. This will do by refrence link given below.

   b. Source Link is: https://nodejs.org/en

   c. First we Install the node js in system then go to CLI and type **node –version** to check the node js version after that type **npm install –g lite-server** to install lite-server so that its display frontend in chrome or any default browser we selected



```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Abdullah Taufiq>node --version
v20.11.0

C:\Users\Abdullah Taufiq>npm install lite-server
[                  ] | idealTree:Abdullah Taufiq: sill idealTree buildDeps
```

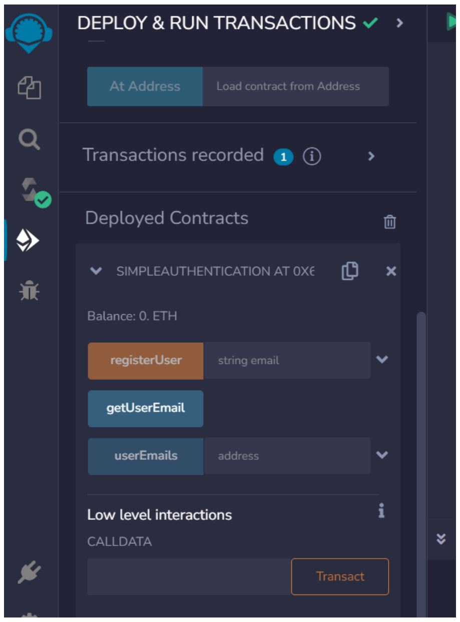
Front End Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Email Registration</title>
    <script
src="https://cdn.jsdelivr.net/npm/web3@1.3.0/dist/web3.min.js"></script>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f7f7f7;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        .container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            text-align: center;
            width: 300px;
        }

        h1 {
            color: #333;
        }

        form {
            margin-top: 20px;
        }

        label {
            font-weight: bold;
            display: block;
            margin-bottom: 8px;
        }

        input {
            width: 100%;
            padding: 10px;
            margin-bottom: 16px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        button {
            background-color: #4caf50;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Email Registration</h1>

    <form id="registrationForm">
        <label for="email">Email:</label>
        <input type="email" id="email" required>
        <br>
        <button type="button" onclick="registerUser()">Register</button>
    </form>

    <button onclick="checkUserEmail()">Check User Email</button>
</div>

<script>
    // Check if Metamask is installed
     if (typeof window.ethereum === 'undefined') {
        alert('Please install Metamask to use this application.');
    }

    // Connect to Metamask
    const web3 = new Web3(window.ethereum);
    const contractAddress = '0xAe7001b7388F15fA9e91c7242B32ABe3b1090fa7'; //
Replace with your deployed smart contract address
    const contractABI =[
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": true,
                "internalType": "address",
                "name": "userAddress",
                "type": "address"
            },
            {
                "indexed": false,
                "internalType": "string",
                "name": "email",
                "type": "string"
            }
        ],
        "name": "UserRegistered",
        "type": "event"
    },
    {
        "inputs": [],
        "name": "getUserEmail",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "email",
                "type": "string"
            }
        ],
        "name": "registerUser",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "name": "userEmails",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];

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

    async function registerUser() {
        const email = document.getElementById('email').value;

        // Simple validation
        if (!email) {
            alert('Email is required.');
            return;
        }

        try {
            // Call the registerUser function in the smart contract
            await simpleAuthContract.methods.registerUser(email).send({ from:
await getAccount() });

            alert('User registered successfully!');
        } catch (error) {
            console.error('Error during registration:', error);
            alert('Error during registration. Please check the console for
details.');
        }
    }

    async function checkUserEmail() {
        try {
            // Call the getUserEmail function in the smart contract
            const userEmail = await
simpleAuthContract.methods.getUserEmail().call({ from: await getAccount() });

            if (userEmail) {
                alert('User email: ' + userEmail);
            } else {
                alert('User not registered or email not provided.');
            }
        } catch (error) {
            console.error('Error checking user email:', error);
            alert('Error checking user email. Please check the console for
details.');
        }
    }

    async function getAccount() {
        const accounts = await window.ethereum.request({ method:
'eth_requestAccounts' });
        return accounts[0];


   }
</script>

</body>
</html>
```

Solidity Code:

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

contract SimpleAuthentication {
    mapping(address => string) public userEmails;

    event UserRegistered(address indexed userAddress, string email);

    function registerUser(string memory email) external {
        require(bytes(userEmails[msg.sender]).length == 0, "User already
registered");
        userEmails[msg.sender] = email;
        emit UserRegistered(msg.sender, email);
    }

    function getUserEmail() external view returns (string memory) {
        return userEmails[msg.sender];
    }
}
```

Outputs:


DEPLOY & RUN TRANSACTIONS

At Address | Load contract from Address

Transactions recorded 1

Deployed Contracts

SIMPLEAUTHENTICATION AT 0XE

Balance: 0. ETH

registerUser | string email

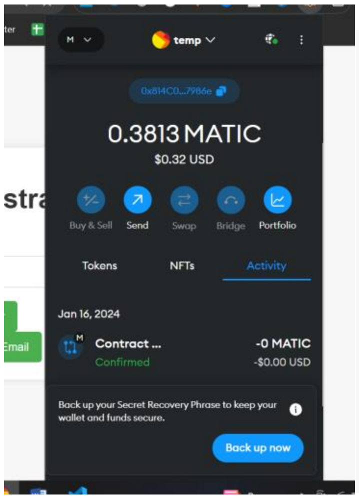
getUserEmail

userEmails | address

Low level interactions

CALLDATA

Transact



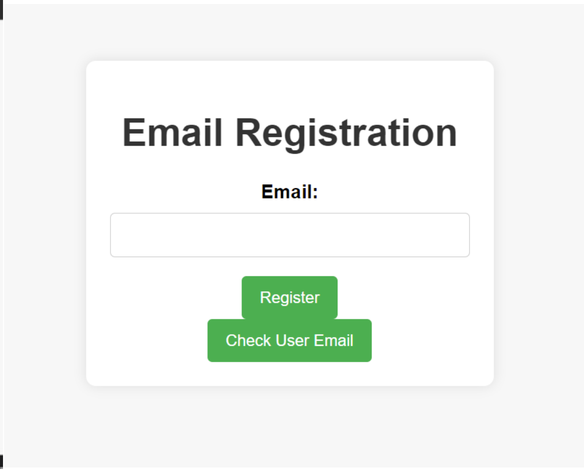
Jan 16, 2024

Contract ... -0 MATIC

Confirmed -$0.00 USD

Back up your Secret Recovery Phrase to keep your wallet and funds secure.

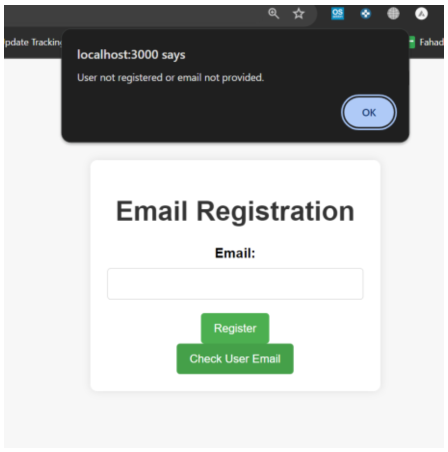
Back up now


Email Registration

Email:

Register

Check User Email


localhost:3000 says

User not registered or email not provided.

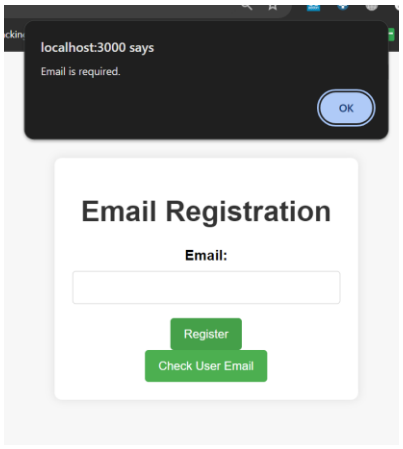
OK

Email Registration

Email:

Register

Check User Email


localhost:3000 says

Email is required.

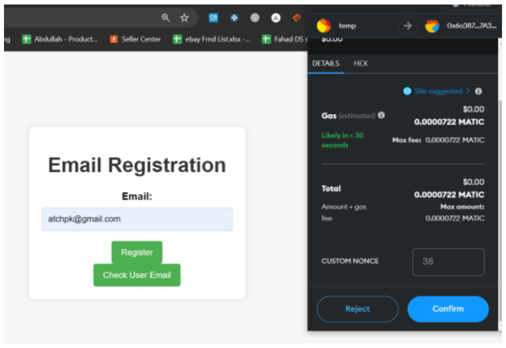
OK

Email Registration

Email:

Register

Check User Email


Email Registration

Email: atchpk@gmail.com

temp → 0x6a387...7A3...

DETAILS HEX

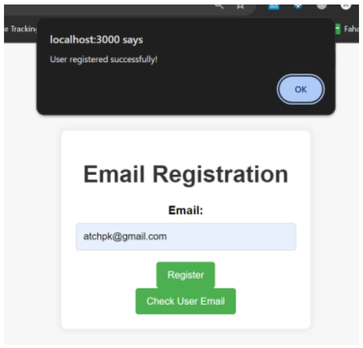
Gas (estimated) $0.00 0.0000722 MATIC

Likely in < 30 seconds Max fee: 0.0000722 MATIC

Total $0.00 0.0000722 MATIC

Amount + gas fee Max amount: 0.0000722 MATIC

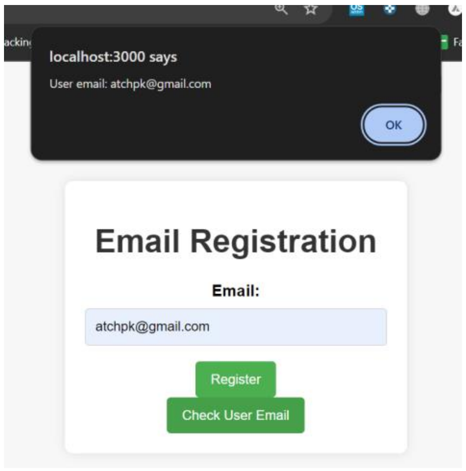
CUSTOM NONCE 38


localhost:3000 says

User registered successfully!

Email Registration

Email: atchpk@gmail.com


localhost:3000 says

User email: atchpk@gmail.com

Email Registration

Email: atchpk@gmail.com

Project Repository

https://drive.google.com/drive/folders/10LQ19HYRlFj2uSFMrZmL_QR6aKZUqjsJ?usp=drive_link

Helping Refrence

https://www.youtube.com/watch?v=1lcLLHzQDCg

Conclusion

This project aims to provide users with a secure and decentralized solution for storing and retrieving email addresses on the blockchain. The integration of Metamask ensures a seamless and user-friendly experience, while the smart contract on the Ethereum blockchain guarantees transparency and immutability.

By the project's completion, user will have a reliable and secure platform for managing his email address, leveraging blockchain technology.

For any further inquiries or clarifications, please feel free to contact us.