Team ID	NM2023TMID00434
Project Name	Ethereum Decentralised Identity Smart Contarct

Project Report Format

1. INTRODUCTION

1.1 Project Overview

Ethereum Decentralised Identity Smart Contract

In an era where digital identities are increasingly valuable yet vulnerable, the "Ethereum Decentralised Identity Smart Contract" emerges as a groundbreaking solution. Leveraging the Ethereum blockchain's capabilities, this smart contract redefines identity management, offering enhanced security, transparency, and user autonomy. Users can securely store their identity details on the Ethereum blockchain, safeguarded by its immutability and cryptographic principles.Unlike traditional systems, this decentralised approach eliminates single points of failure and minimises the risk of data breaches. Moreover, it empowers individuals with control over their own information, enabling selective sharing and revocation of access. The Ethereum Decentralised Identity Smart Contract represents a paradigm shift in identity management, where trust is established through blockchain consensus and verification occurs end-to-end, enhancing security and reducing fraud. This innovation not only ensures the integrity of personal information but also opens doors to a wide range of applications, from secure voting systems to streamlined financial transactions, all underpinned by a resilient and transparent identity management framework.

1.2 Purpose

The purpose of Ethereum decentralized identity smart contracts is to provide a secure and user-centric way to manage and represent individuals' identities in a decentralized manner on the Ethereum blockchain.

2. LITERATURE SURVEY

2.1 Existing problem

While Ethereum-based decentralized identity (DID) and self-sovereign identity (SSI) systems offer significant benefits, they also face several challenges and problems. Some of the existing issues in Ethereum decentralized identity smart contracts

2.2 References

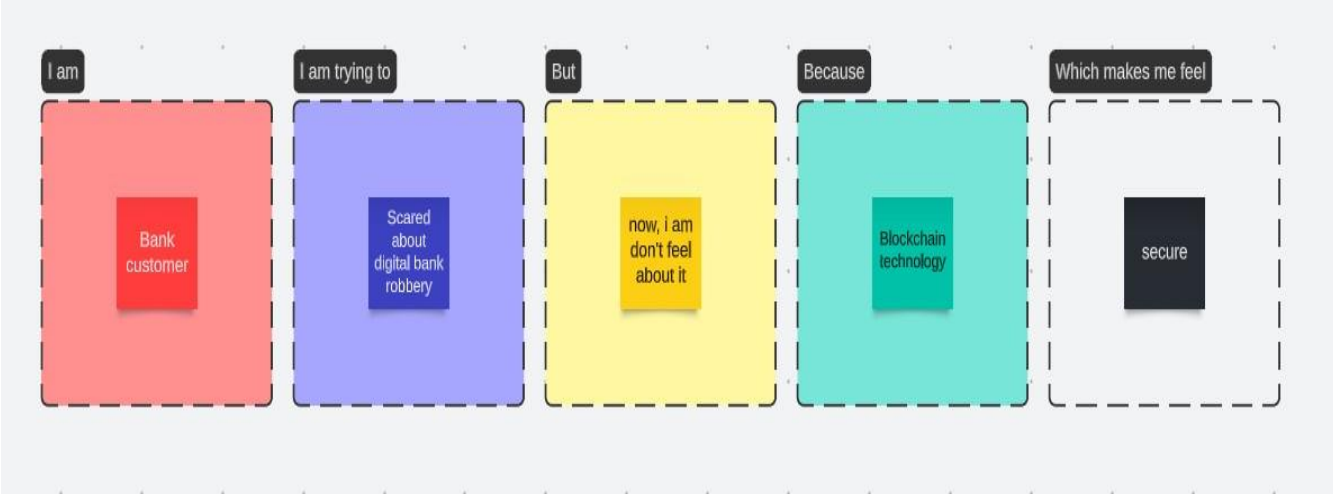
1. Ethereum Improvement Proposals (EIPs): Search for Ethereum Improvement Proposals related to identity. These proposals often contain technical details and discussions related to implementing decentralized identity solutions on Ethereum.
2. W3C Decentralized Identity Working Group: The World Wide Web

Consortium (W3C) has a Decentralized Identity Working Group that develops standards and specifications for decentralized identity. Their work can provide valuable insights and references.

3. Ethereum Developer Documentation: Explore the Ethereum developer documentation, including resources on smart contracts and blockchain development, which may cover aspects related to decentralized identity.
4. Decentralized Identity Communities: Participate in forums, discussions, and communities that focus on decentralized identity, such as those on GitHub, Reddit, or dedicated forums for Ethereum developers and blockchain enthusiasts.
5. Research Papers and Articles: Academic research papers and articles often provide insights into the technical aspects of decentralized identity and may reference specific implementations or smart contract code.

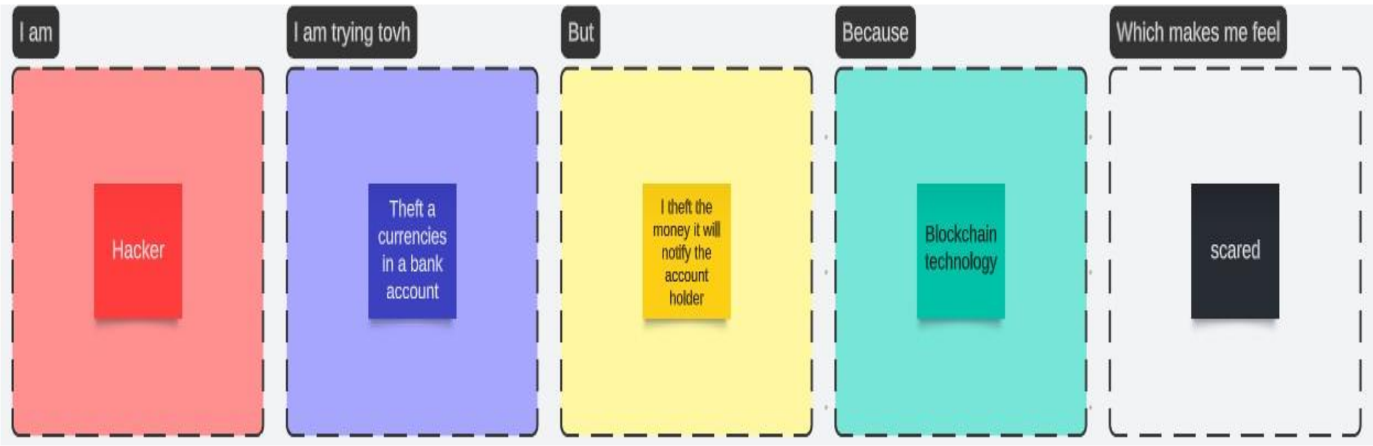
2.3problem Statement Definition

Problem statement 1:



I am	I am trying to	But	Because	Which makes me feel
Bank customer	Scared about digital bank robbery	now, i am don't feel about it	Blockchain technology	secure

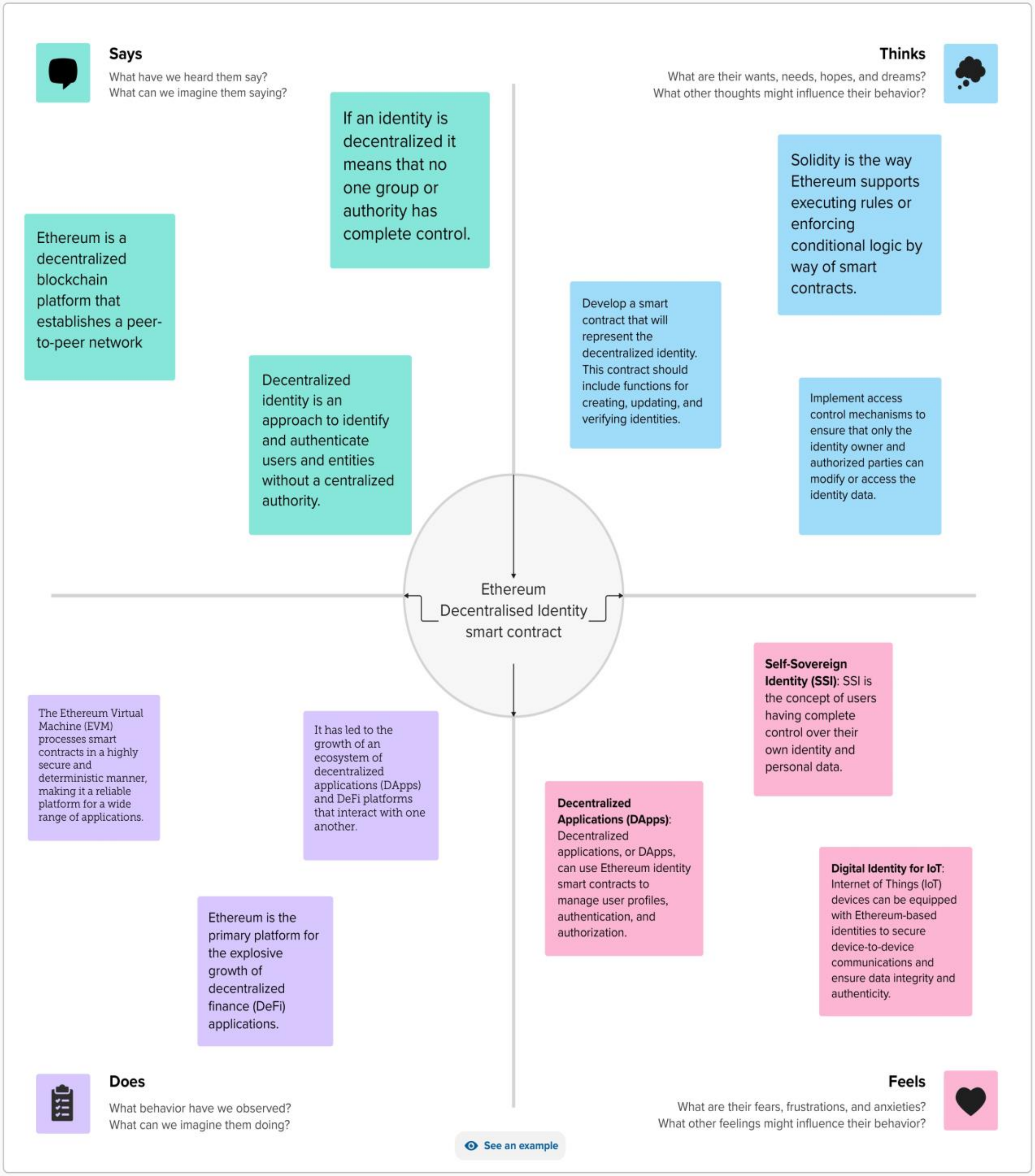
Problem statement 2:



I am	I am trying tovh	But	Because	Which makes me feel
Hacker	Theft a currencies in a bank account	I theft the money it will notify the account holder	Blockchain technology	scared

3. IDEATION & PROPOSED SOLUTION

3.1Empathy Map Canvas



Says

What have we heard them say?
What can we imagine them saying?

If an identity is decentralized it means that no one group or authority has complete control.

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network

Decentralized identity is an approach to identify and authenticate users and entities without a centralized authority.

Thinks

What are their wants, needs, hopes, and dreams?
What other thoughts might influence their behavior?

Solidity is the way Ethereum supports executing rules or enforcing conditional logic by way of smart contracts.

Develop a smart contract that will represent the decentralized identity. This contract should include functions for creating, updating, and verifying identities.

Implement access control mechanisms to ensure that only the identity owner and authorized parties can modify or access the identity data.

Ethereum Decentralised Identity smart contract

The Ethereum Virtual Machine (EVM) processes smart contracts in a highly secure and deterministic manner, making it a reliable platform for a wide range of applications.

It has led to the growth of an ecosystem of decentralized applications (DApps) and DeFi platforms that interact with one another.

Ethereum is the primary platform for the explosive growth of decentralized finance (DeFi) applications.

Self-Sovereign Identity (SSI): SSI is the concept of users having complete control over their own identity and personal data.

Decentralized Applications (DApps): Decentralized applications, or DApps, can use Ethereum identity smart contracts to manage user profiles, authentication, and authorization.

Digital Identity for IoT: Internet of Things (IoT) devices can be equipped with Ethereum-based identities to secure device-to-device communications and ensure data integrity and authenticity.

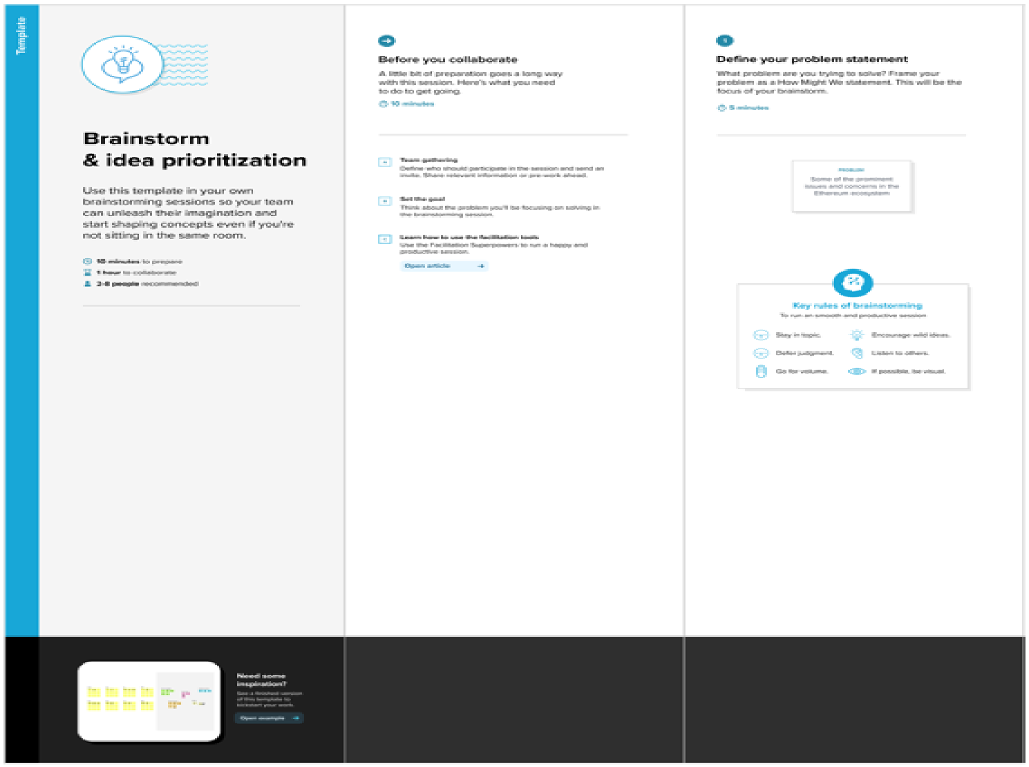
Does

What behavior have we observed?
What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties?
What other feelings might influence their behavior?

3.2 Ideation & Brainstorming



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

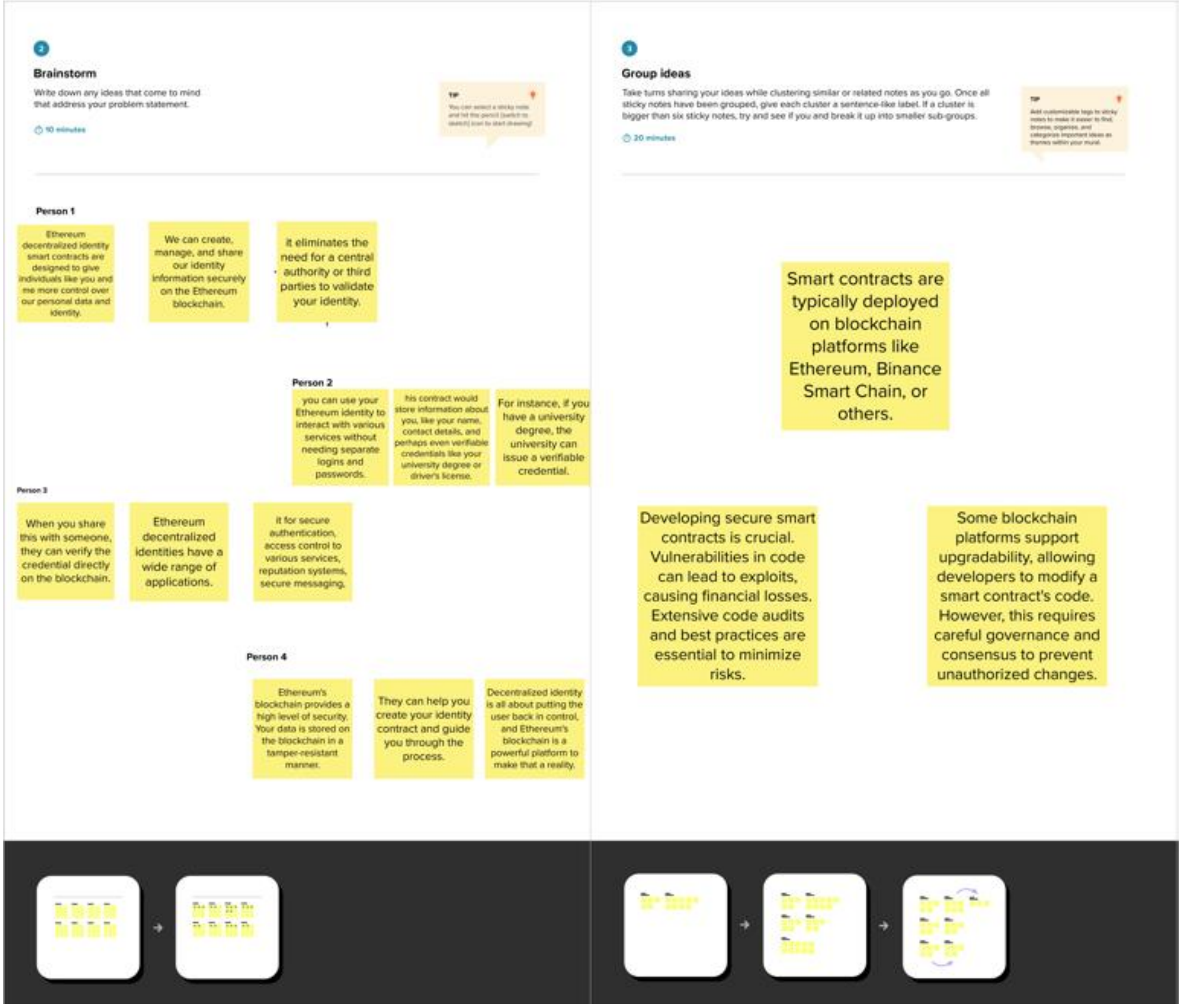
5 minutes

PROBLEM
Some of the prominent issues and concerns in the Ethereum ecosystem

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Person 1

Ethereum decentralized identity smart contracts are designed to give individuals like you and me more control over our personal data and identity.

We can create, manage, and share our identity information securely on the Ethereum blockchain.

it eliminates the need for a central authority or third parties to validate your identity.

Person 2

you can use your Ethereum identity to interact with various services without needing separate logins and passwords.

his contract would store information about you, like your name, contact details, and perhaps even verifiable credentials like your university degree or driver's license.

For instance, if you have a university degree, the university can issue a verifiable credential.

Person 3

When you share this with someone, they can verify the credential directly on the blockchain.

Ethereum decentralized identities have a wide range of applications.

it for secure authentication, access control to various services, reputation systems, secure messaging.

Person 4

Ethereum's blockchain provides a high level of security. Your data is stored on the blockchain in a tamper-resistant manner.

They can help you create your identity contract and guide you through the process.

Decentralized identity is all about putting the user back in control, and Ethereum's blockchain is a powerful platform to make that a reality.

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

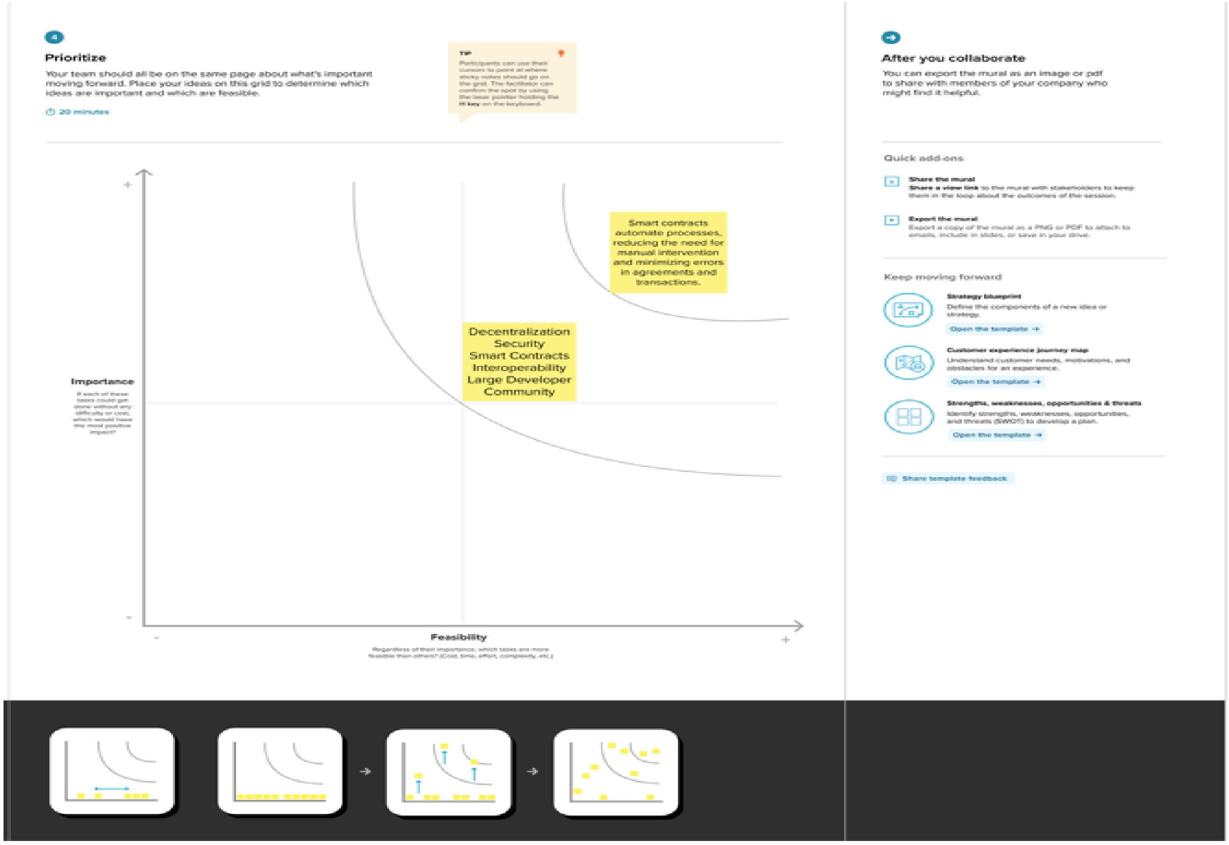
TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Smart contracts are typically deployed on blockchain platforms like Ethereum, Binance Smart Chain, or others.

Developing secure smart contracts is crucial. Vulnerabilities in code can lead to exploits, causing financial losses. Extensive code audits and best practices are essential to minimize risks.

Some blockchain platforms support upgradability, allowing developers to modify a smart contract's code. However, this requires careful governance and consensus to prevent unauthorized changes.



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Smart contracts automate processes, reducing the need for manual intervention and minimizing errors in agreements and transactions.

Decentralization
Security
Smart Contracts
Interoperability
Large Developer
Community

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

4. REQUIREMENT ANALYSIS

4.1Functional requirement

Functional requirements for Ethereum decentralized identity smart contracts typically specify the capabilities and features that the smart contract should possess to effectively manage decentralized identities on the Ethereum blockchain.

1. Cross-Platform Compatibility: Ensure that the decentralized identity system is compatible with various platforms and devices, making it accessible and functional for users on different devices and environments.

2. Identity Creation: Users should be able to create a new decentralized identity, including defining attributes like name, contact information, and other relevant personal data.

3. Key Pair Management: The smart contract must handle the creation, storage, and management of cryptographic key pairs for the user. This includes generating and securing public and private keys.

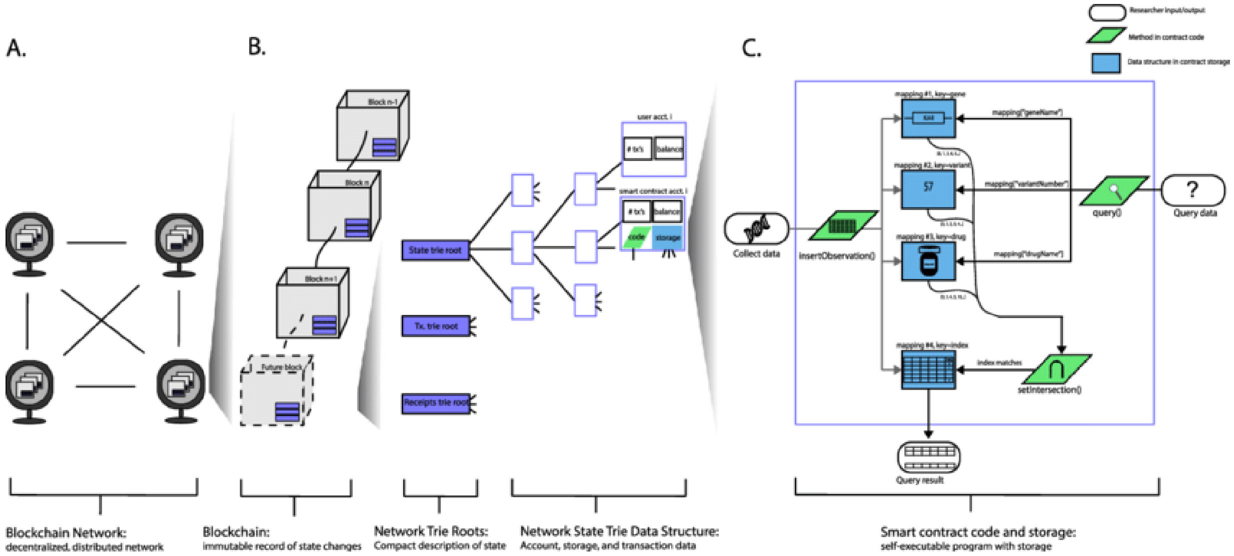
4. Identity Verification: Users should be able to request and receive verifiable credentials from trusted sources, such as educational institutions, government agencies, or banks

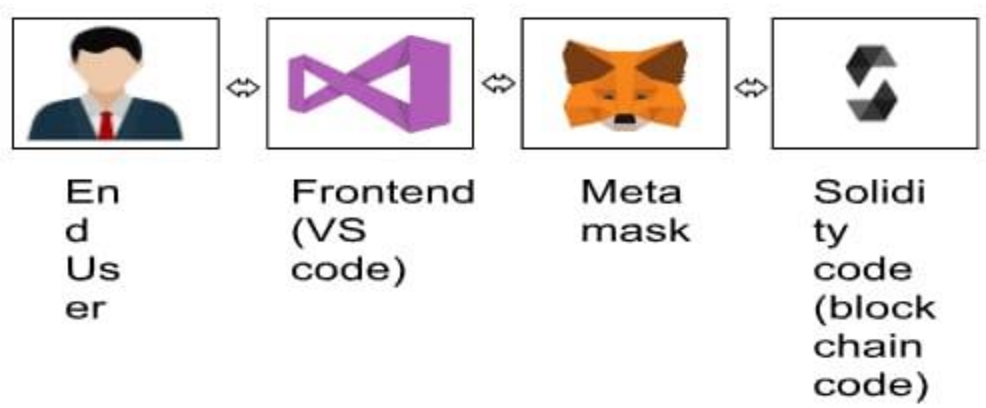
4.2 Non-Functional requirement

Non-functional requirements for Ethereum decentralized identity smart contracts encompass qualities or characteristics of the system beyond its core functionality. These requirements focus on aspects such as performance, security, usability, and reliability.

5. PROJECT DESIGN

5.1Data Flow Diagrams & User Stories

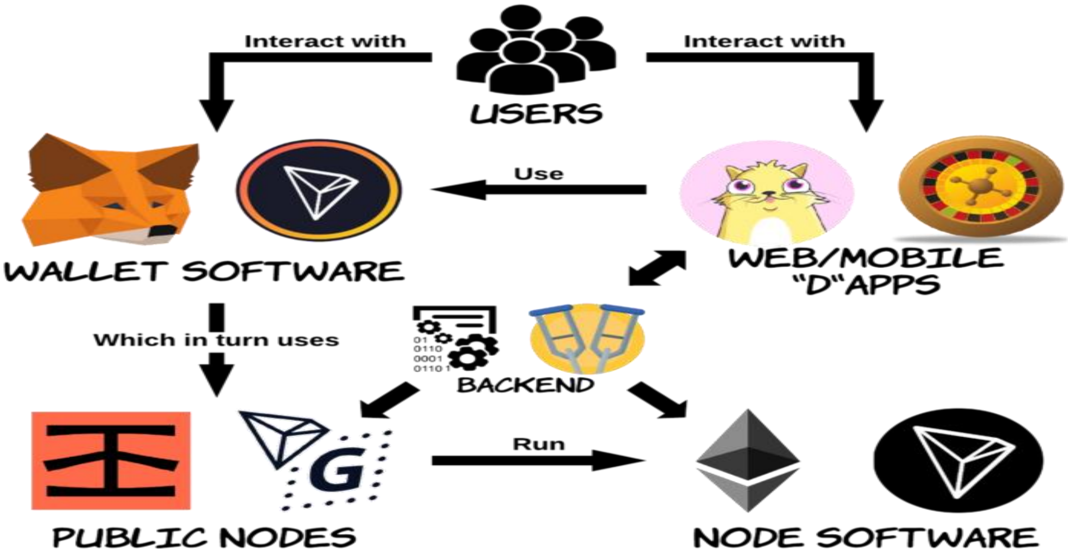




User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Team Member
New User	User Registration	US01	As a new user, I want to create a decentralized identity on Ethereum to control my personal information securely. .	• The user can complete the registration process. • The user's decentralized identity is stored on the Ethereum blockchain. • The user receives their unique Decentralized Identifier (DID).	High	G.GOKULRAJ
Identity Owner	Identity Update	US02	As a user, I want to update my decentralized identity with new information, such as a change of address or a new email.	• The user can access and edit their identity information. • Changes are securely recorded on the blockchain. • The user's private key is required to initiate updates.	Medium	P. Kumaravel

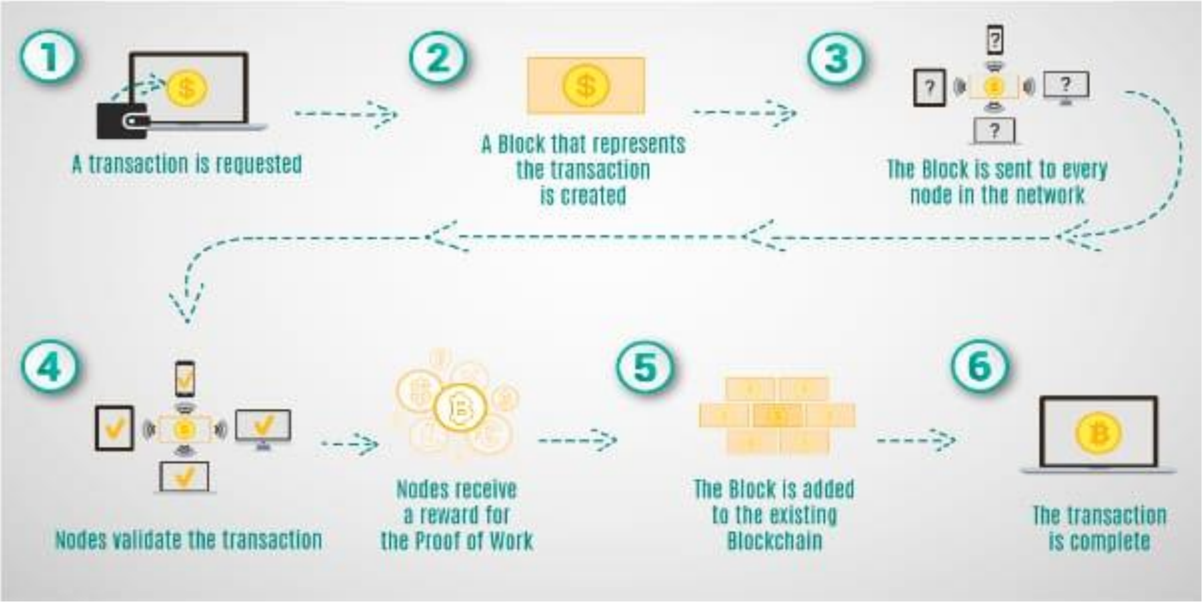
Relying Party (Third Party)	Credential Validation	US03	As a relying party, I want to verify the authenticity of verifiable credentials presented by a user to ensure the integrity of identity-related transactions.	• The relying party can access the user's presented verifiable credentials. • The credentials are securely verified. • The relying party can trust the validity of the credentials for the transaction.	High	K.SIVASAKTHI
Identity Owner	Privacy Preferences	US04	As a user, I want to set privacy preferences for my identity to control how much information is shared during transactions and interactions.	• The user can access and configure privacy settings. • Privacy settings are clearly defined and user-friendly. • The user can restrict or allow the sharing of specific attributes.	Medium	P. Sneha

5.2Solution Architecture



6. PROJECT PLANNING & SCHEDULING

Technical Architecture


1
A transaction is requested
2
A Block that represents the transaction is created
3
The Block is sent to every node in the network
4
Nodes validate the transaction
Nodes receive a reward for the Proof of Work
5
The Block is added to the existing Blockchain
6
The transaction is complete

6.1Sprint Planning & Estimation

1. Define the Scope:

Clearly define the features and functionality you want in your DID smart contract. For Ethereum, this typically includes managing identity claims, public keys, and the associated data in a decentralized and secure manner.

2. User Stories and Epics:

Break down the project into user stories and epics. For example, user stories may include user registration, claim issuance, claim verification, and claim revocation.

3. Story Estimation:

Estimate the effort required for each user story or epic. You can use story points or time-based estimates (e.g., in hours) for this purpose.

4. Prioritization:

Prioritize the user stories based on their importance and dependencies. Identify the minimum viable product (MVP) features that should be implemented first.

5. Sprint Planning:

Determine the sprint length (e.g., 2 weeks) and select a set of user stories to be completed during the sprint. These stories should align with your project's objectives and priorities.

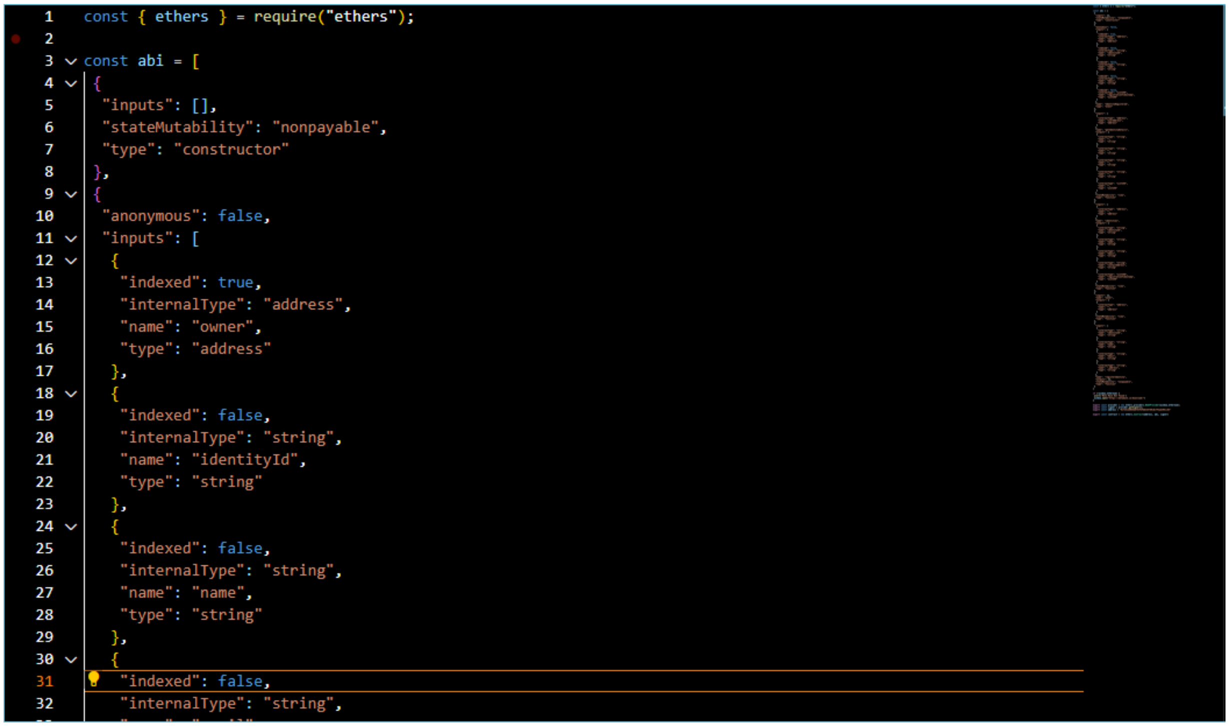
6.2Sprint Delivery Schedule

Creating a sprint delivery schedule for an Ethereum decentralized identity (DID) smart contract project involves breaking down the project into manageable tasks and allocating them to specific sprints. A typical sprint duration is two weeks, but it can vary based on your team's preferences and project complexity.

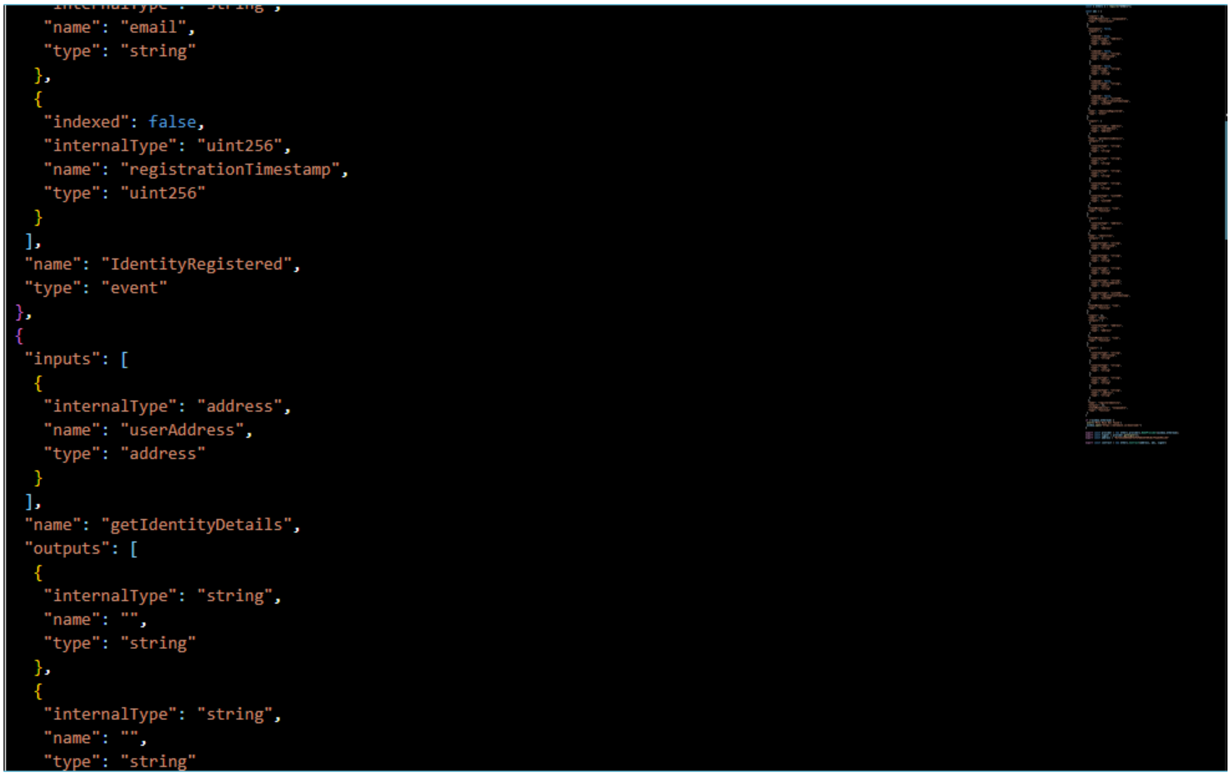
7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

Ethereum decentralized identity (DID) smart contracts are designed to provide users with control over their digital identities and personal data while ensuring security and privacy.



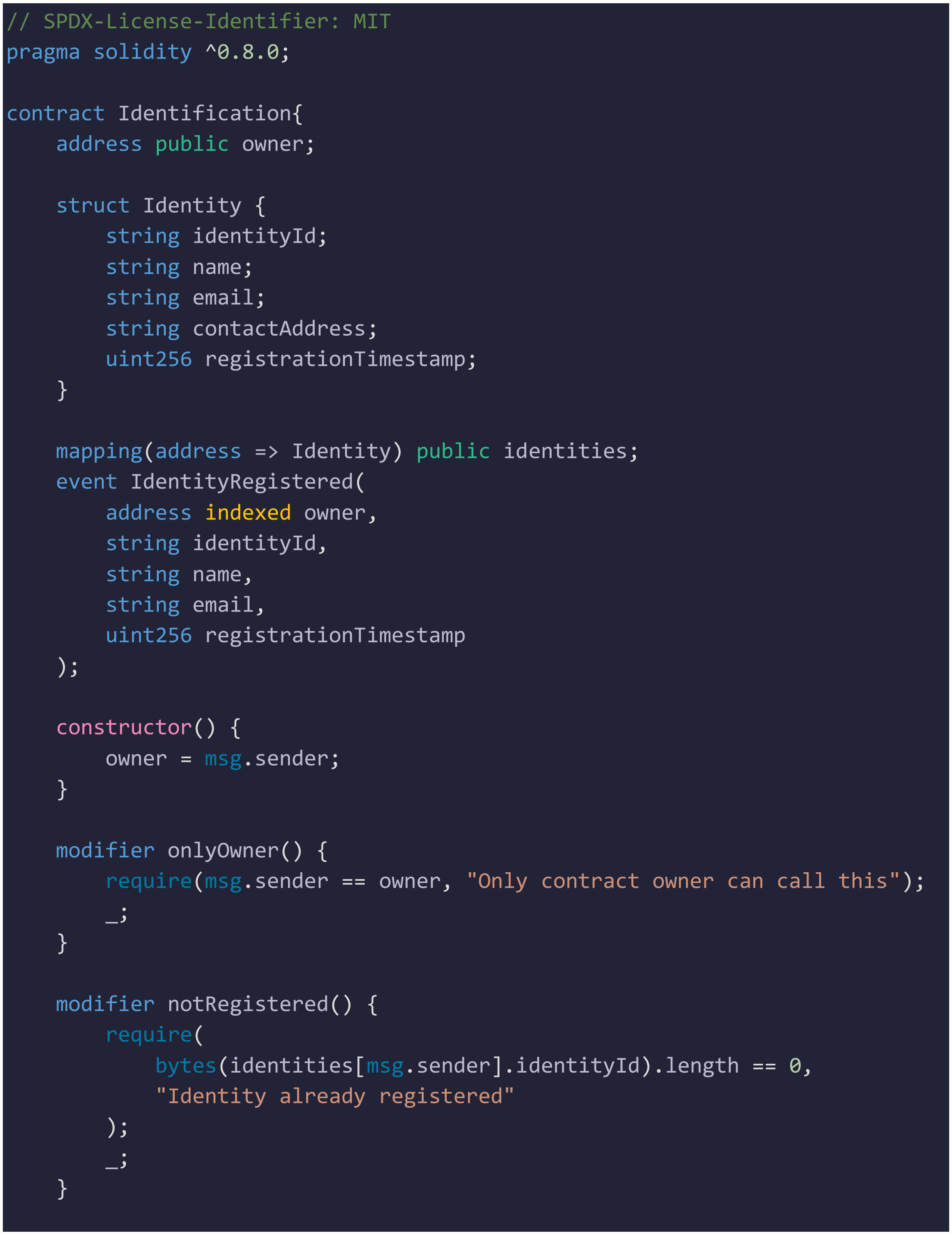
```
const { ethers } = require("ethers");

const abi = [
 {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "address",
    "name": "owner",
    "type": "address"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "identityId",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "name",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
```



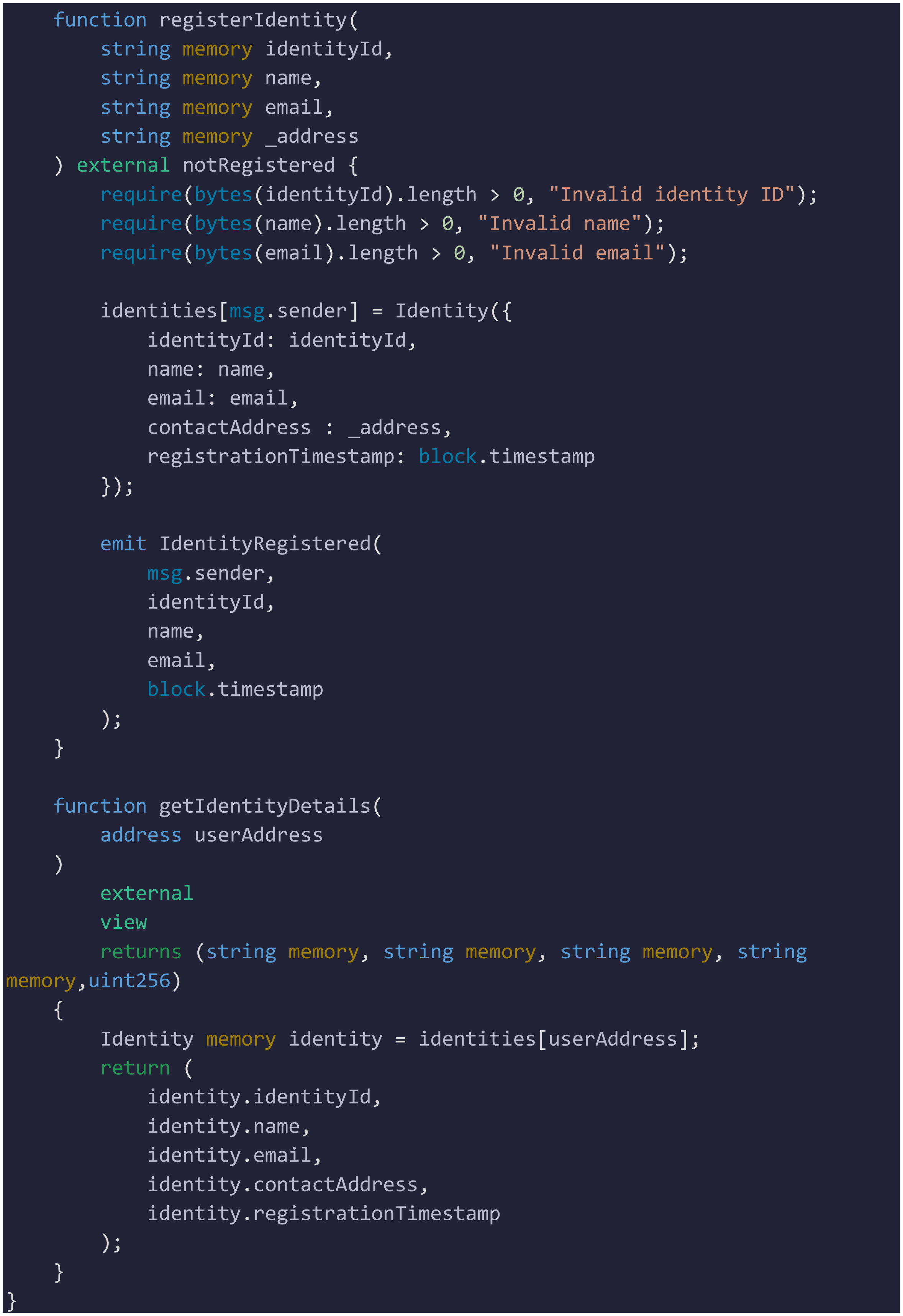
```
    "internalType": "string",
    "name": "email",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "uint256",
    "name": "registrationTimestamp",
    "type": "uint256"
   }
  ],
  "name": "IdentityRegistered",
  "type": "event"
 },
 {
  "inputs": [
   {
    "internalType": "address",
    "name": "userAddress",
    "type": "address"
   }
  ],
  "name": "getIdentityDetails",
  "outputs": [
   {
    "internalType": "string",
    "name": "",
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "",
    "type": "string"
```

7.2Feature 2

Ethereum DID smart contracts are a fundamental component of decentralized identity systems in the Web3 ecosystem, enabling users to have greater control over their digital identities, data, and privacy. The specific features and design of these contracts can vary depending on the project's goals and requirements.



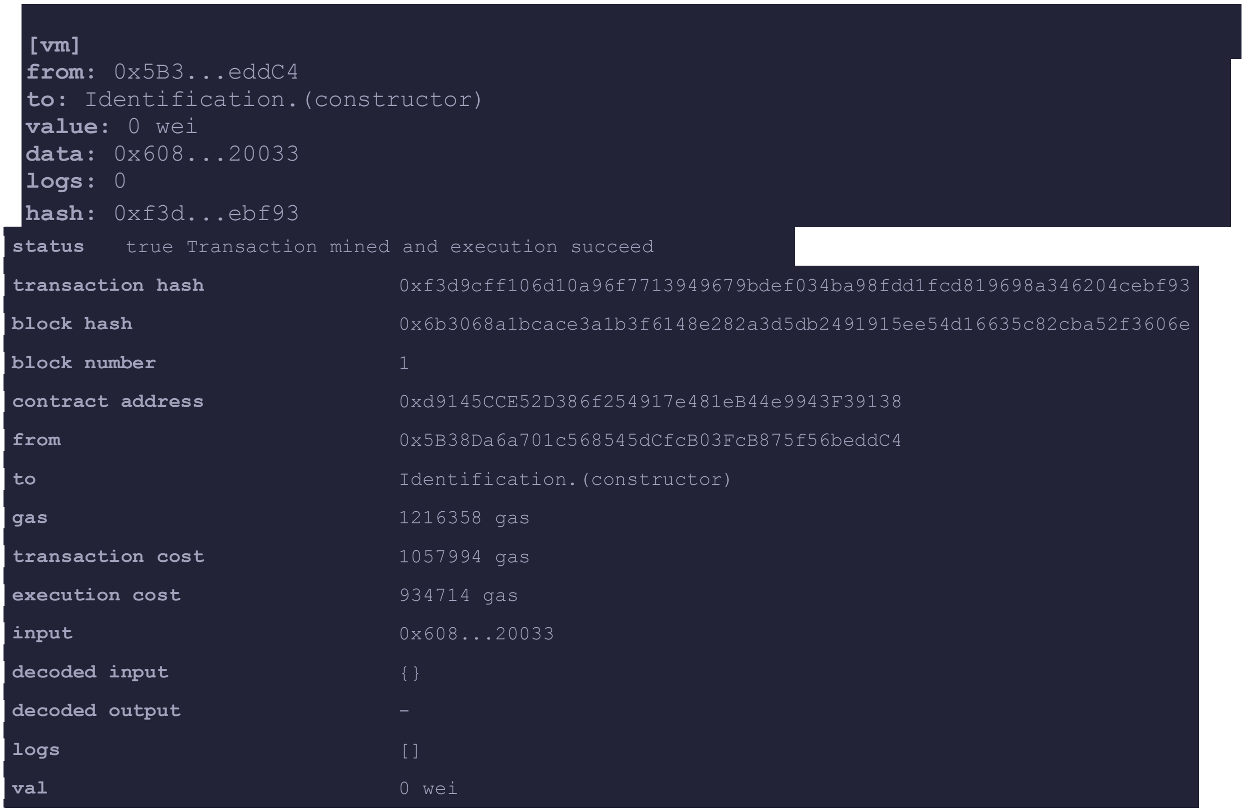
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Identification{
    address public owner;

    struct Identity {
        string identityId;
        string name;
        string email;
        string contactAddress;
        uint256 registrationTimestamp;
    }

    mapping(address => Identity) public identities;
    event IdentityRegistered(
        address indexed owner,
        string identityId,
        string name,
        string email,
        uint256 registrationTimestamp
    );

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only contract owner can call this");
        _;
    }

    modifier notRegistered() {
        require(
            bytes(identities[msg.sender].identityId).length == 0,
            "Identity already registered"
        );
        _;
    }
```



```
    function registerIdentity(
        string memory identityId,
        string memory name,
        string memory email,
        string memory _address
    ) external notRegistered {
        require(bytes(identityId).length > 0, "Invalid identity ID");
        require(bytes(name).length > 0, "Invalid name");
        require(bytes(email).length > 0, "Invalid email");

        identities[msg.sender] = Identity({
            identityId: identityId,
            name: name,
            email: email,
            contactAddress : _address,
            registrationTimestamp: block.timestamp
        });

        emit IdentityRegistered(
            msg.sender,
            identityId,
            name,
            email,
            block.timestamp
        );
    }

    function getIdentityDetails(
        address userAddress
    )
        external
        view
        returns (string memory, string memory, string memory, string
memory,uint256)
    {
        Identity memory identity = identities[userAddress];
        return (
            identity.identityId,
            identity.name,
            identity.email,
            identity.contactAddress,
            identity.registrationTimestamp
        );
    }
}
```

8. PERFORMANCE TESTING

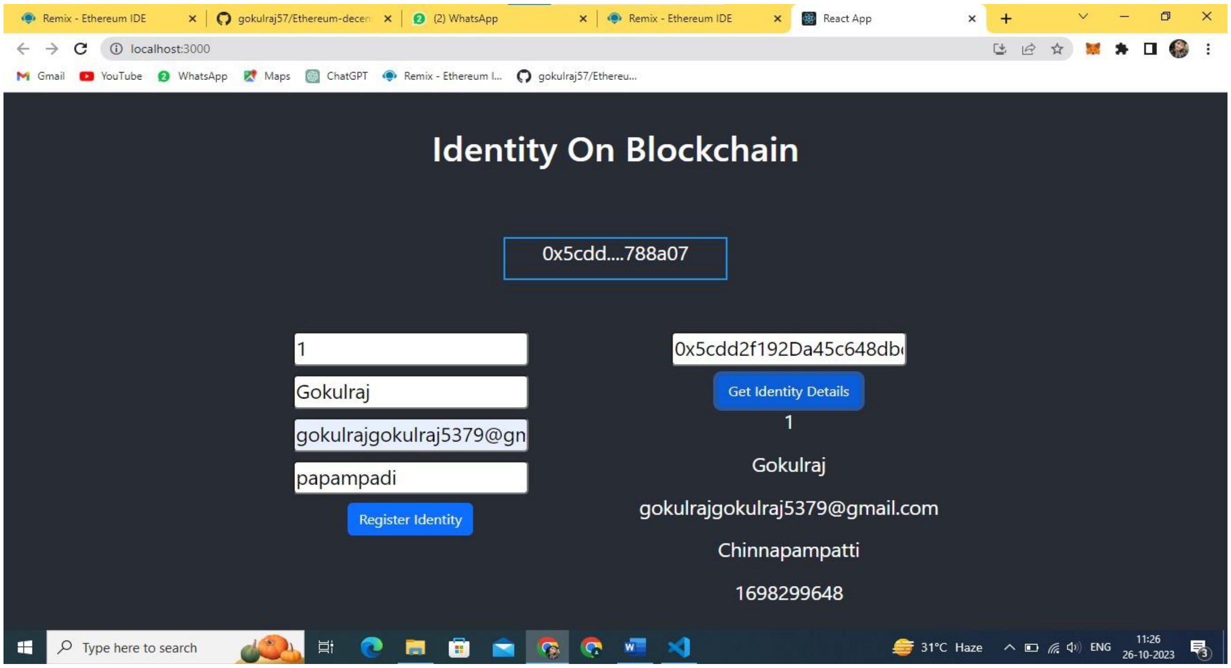
8.1Performace Metrics



```
[vm]
from: 0x5B3...eddC4
to: Identification.(constructor)
value: 0 wei
data: 0x608...20033
logs: 0
hash: 0xf3d...ebf93
status    true Transaction mined and execution succeed
transaction hash                 0xf3d9cff106d10a96f7713949679bdef034ba98fdd1fcd819698a346204cebf93
block hash                       0x6b3068a1bcace3a1b3f6148e282a3d5db2491915ee54d16635c82cba52f3606e
block number                     1
contract address                 0xd9145CCE52D386f254917e481eB44e9943F39138
from                             0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                               Identification.(constructor)
gas                              1216358 gas
transaction cost                 1057994 gas
execution cost                   934714 gas
input                            0x608...20033
decoded input                    {}
decoded output                   -
logs                             []
val                              0 wei
```

9. RESULTS

Output Screenshots



Identity On Blockchain

0x5cdd....788a07

1

Gokulraj

gokulrajgokulraj5379@gn

papampadi

Register Identity

0x5cdd2f192Da45c648db

Get Identity Details

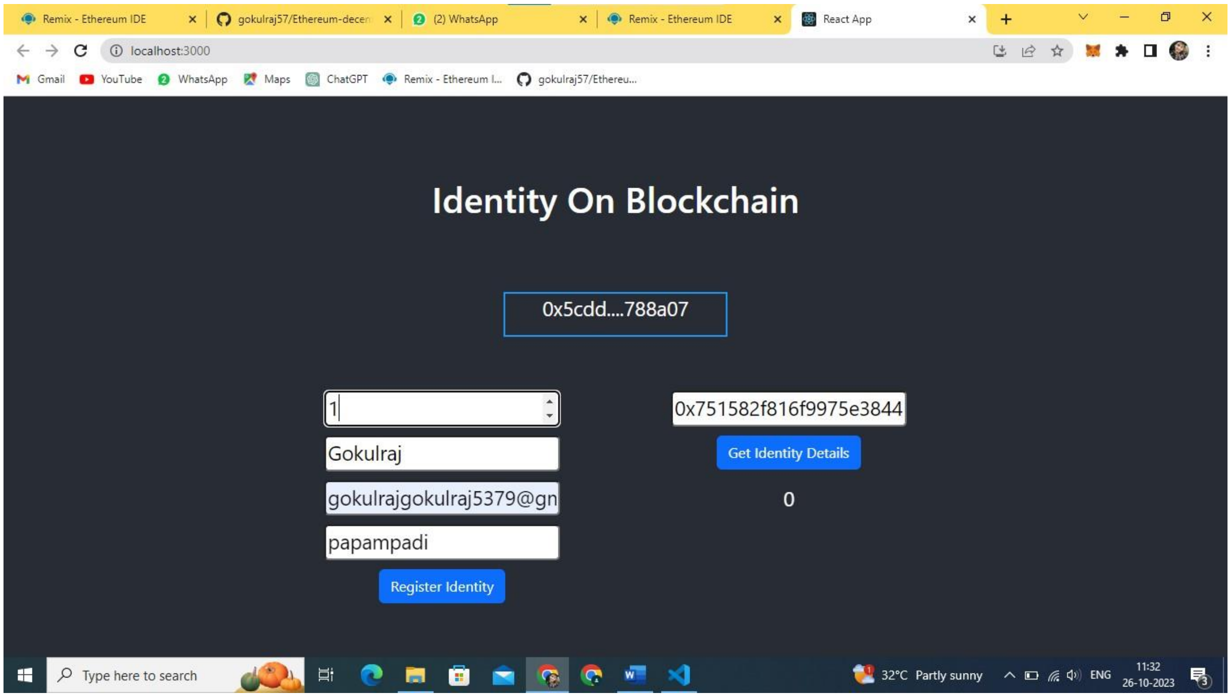
1

Gokulraj

gokulrajgokulraj5379@gmail.com

Chinnapampatti

1698299648



Identity On Blockchain

0x5cdd....788a07

1

Gokulraj

gokulrajgokulraj5379@gn

papampadi

0x751582f816f9975e3844

0

10.ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- User Control:

Users have complete control over their digital identities. They can create, update, and manage their decentralized identities and associated claims.

- Privacy:

Users can selectively disclose information, enhancing privacy. They only share the information they choose, reducing the risk of overexposing personal data.

- Security:

Ethereum DID smart contracts utilize strong cryptographic methods, making it difficult for malicious actors to tamper with or forge identities and claims.

- Immutability:

Once recorded on the Ethereum blockchain, DIDs and claims are immutable and tamper-resistant, providing a high level of trust in the data.

DISADVANTAGES:

- Complexity: Implementing and managing Ethereum DID smart contracts can be complex, requiring a deep understanding of blockchain technology, cryptography, and identity standards.
- Scalability: The Ethereum blockchain has limitations in terms of scalability. As more users adopt DID solutions, network congestion and high gas fees can become issues.
- Smart Contract Vulnerabilities: Smart contracts can have vulnerabilities that might be exploited by malicious actors if not properly audited and secured.

11.CONCLUSION

In conclusion, Ethereum decentralized identity (DID) smart contracts represent a promising and innovative approach to digital identity management, providing individuals with greater control, security, and privacy over their personal data. While there are advantages to using Ethereum DID smart contracts, including user empowerment, enhanced security, and privacy, there are also challenges and considerations that need to be addressed.

12.FUTURE SCOPE

The future of Ethereum decentralized identity smart contracts is closely tied to the broader evolution of blockchain technology, digital identity, and the growing emphasis on user sovereignty and data privacy. As these trends continue to develop, decentralized identity solutions are poised to play a significant role in shaping the digital landscape.

13.APPENDIX

Source Code

Solidity code:

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Identification{

  address public owner;

  struct Identity {

    string identityId;
    string name;
    string email;
    string contactAddress;
    uint256 registrationTimestamp;
}
mapping(address => Identity) public identities;
event IdentityRegistered(
    address indexed owner,
    string identityId,
    string name,
    string email,
    uint256 registrationTimestamp
);
    constructor() {
    owner = msg.sender;
}
modifier onlyOwner() {
    require(msg.sender == owner, "Only contract owner can call this");
    _;
}
modifier notRegistered() {
    require(
        bytes(identities[msg.sender].identityId).length == 0,
        "Identity already registered"
    );
    _;
}
function registerIdentity(
  string memory identityId,
  string memory name,
  string memory email,
  string memory _address
) external notRegistered {
  require(bytes(identityId).length > 0, "Invalid identity ID");
  require(bytes(name).length > 0, "Invalid name");
  require(bytes(email).length > 0, "Invalid email");
  identities[msg.sender] = Identity({
    identityId: identityId,
    name: name,
    email: email,
    contactAddress : _address,
    registrationTimestamp: block.timestamp
  });
  emit IdentityRegistered(
    msg.sender,
    identityId,
    name,
    email,
    block.timestamp
  );
}
function getIdentityDetails(
  address userAddress
)
  external
  view
```

```
    returns (string memory, string memory, string memory, string memory,uint256)
  {
    Identity memory identity = identities[userAddress];
    return (
      identity.identityId,
      identity.name,
      identity.email,
      identity.contactAddress,
      identity.registrationTimestamp
    );
  }
}
```

VS Code:

```
const { ethers } = require("ethers");

const abi = [
 {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "address",
 "name": "owner",
 "type": "address"
},
{
 "indexed": false,
 "internalType": "string",
 "name": "identityId",
 "type": "string"
},
{
 "indexed": false,
 "internalType": "string",
 "name": "name",
 "type": "string"
},
{
 "indexed": false,
 "internalType": "string",
 "name": "email",
 "type": "string"
},
{
 "indexed": false,
 "internalType": "uint256",
 "name": "registrationTimestamp",
 "type": "uint256"
}
],
 "name": "IdentityRegistered",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "address",
   "name": "userAddress",
   "type": "address"
  }
 ],
 "name": "getIdentityDetails",
 "outputs": [
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "name": "identities",
 "outputs": [
  {
   "internalType": "string",
   "name": "identityId",
   "type": "string"
   },
   {
    "internalType": "string",
    "name": "name",
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "email",
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "contactAddress",
    "type": "string"
   },
   {
    "internalType": "uint256",
    "name": "registrationTimestamp",
    "type": "uint256"
   }
  ],
  "stateMutability": "view",
  "type": "function"
 },
 {
  "inputs": [],
  "name": "owner",
  "outputs": [
   {
    "internalType": "address",
    "name": "",
    "type": "address"
   }
  ],
  "stateMutability": "view",
  "type": "function"
 },
 {
  "inputs": [
   {
    "internalType": "string",
    "name": "identityId",
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "name",
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "email",
    "type": "string"
   },
   {
    "internalType": "string",

    "name": "_address",

    "type": "string"

   }

  ],

  "name": "registerIdentity",

  "outputs": [],

  "stateMutability": "nonpayable",

  "type": "function"

 }

]


if (!window.ethereum) {

 alert('Meta Mask Not Found')

 window.open("https://metamask.io/download/")

}

export const provider = new ethers.providers.Web3Provider(window.ethereum);

export const signer = provider.getSigner();

export const address = "0xf241e6055b8f4CF2Fb8C35769cBc7F2aAC95ccE3"


export const contract = new ethers.Contract(address, abi, signer
```

GitHub & Project Demo Link

GitHub:

https://github.com/gokulraj57/Ethereum-decentralised-identity-smart-contarct