Date	21.10.2023
Team id	NM2023TMID01519
Project name	Agriculture Docs Chain

INDEX

11. CONCLUSION

12. FUTURESCOPE

AGRICULTUREDOCSCHAIN

1.   INTRODUCTION

PROJECTOVERVIEW

The AgricultureDocsChainisaninnovativeendeavordesignedtostreamlineand modernize the management of agriculture data through the utilization of blockchain technology. At   itscore,thisprojectcentersaroundthedevelopmentofasmartcontract ontheEthereumblockchain, meticulouslycrafted to store, query, andupdateessential agricultural information. The technical stack employed in this project is robust, with endusers,auserfriendlyfrontenddevelopedinVisualStudioCode,andtheintegration of the Metamask wallet for seamless interaction with the Ethereum blockchain. The smartcontract,meticulouslyconstructedusingRemixIDEandEthereum,standsasthe keystoneofthisproject, facilitatingthesecurestorageand managementofagricultural data. To enhance user experience and accessibility, the frontend boasts a straightforward interface, featuring buttons for actions such as connecting the wallet, addingnewproductentries,updatingexistingdata,andretrievingproductdetails.Users can input product information, including product ID, name, description, and quantity, withease.Thisprojectaddressesthepressingneedforefficientandsecureagricultural   data management, providing a solid foundation for future enhancements, including additional features, heightened security measures, and more intricate access control mechanisms. In practice, this "Agriculture Docs Chain " aims to revolutionize the agriculture industry by delivering a robust and user-friendly platform for data management and decision making, potentially transforming how farmers and stakeholders interact with agricultural data.

PURPOSE

The primary purpose of the Agriculture Docs Chain is to address the longstanding challenges associated with the management of agricultural data. It seeks to empower farmers and stakeholders with a cutting edge solution that leverages the Ethereum blockchain, therebyrevolutionizing the way vital agricultural information is stored and accessed.Bycraftingasmartcontractthatallowsfortheaddition,querying,andupdating ofdata,theprojectaimstostreamlinetheentireprocess,simplifyingitforendusers.This project serves to bridge the gap between intricate business problems in agriculture and the technological solutions that can effectively resolve them. It seeks to ensure data privacy, integrity, and security, promoting trust and transparency among users. Furthermore,thepurposeextendstoenhancingdecisionmakingintheagriculturesector by providing real time access to valuable insights, such as crop yields, weather conditions,andmarkettrends.Inessence,the"AgricultureDocsChain"projectisdriven bythepurposeofadvancingtheagriculturalindustryintoaneweraofdatamanagement. Itpromisestodeliverauser-friendlyplatformthatnotonlymakesdataentryandretrieval a breeze but also paves the way for additional features and stringent security measures, setting the stage for a fully functionaland secure agriculturaldata management system.

2.   EXISTINGSYSTEM
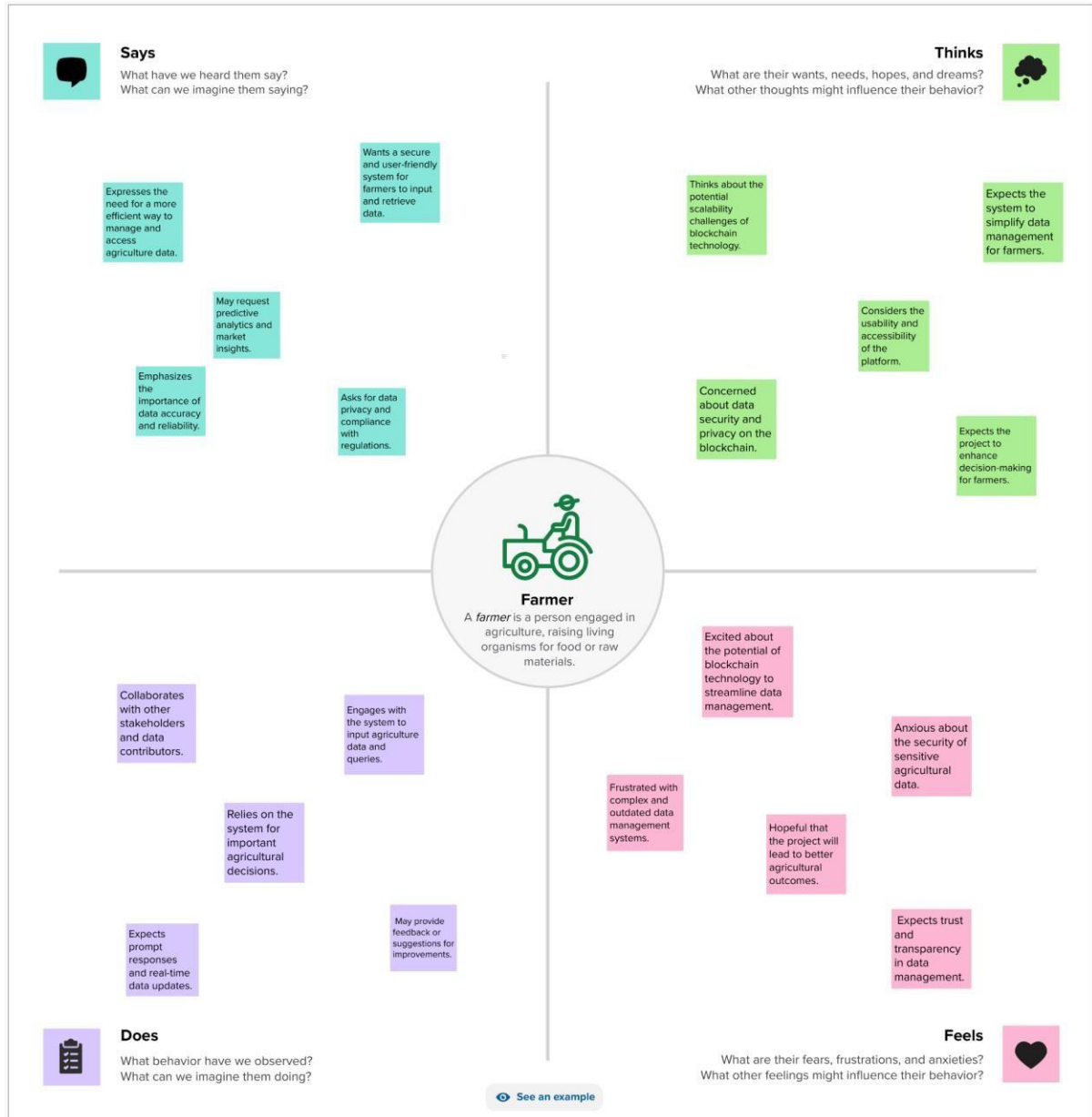
EXISTINGPROBLEM

The existing problem within the agricultural sector lies in the antiquated and often inefficient methods of data management. Traditional paper based records and scattered digital documents make it challenging for farmers and stakeholders to access, update, and analyze crucial agricultural information. This not only results in time consuming processes but also increases the risk of errors, hindering informed decision making. Moreover, data security and privacy are frequently compromised in these outdated systems. The "Agriculture Docs Chain" project addresses this issue by introducing a modern, blockchain based solution that streamlines data management, enhances accessibility, and prioritizes data security, ultimately revolutionizing how agricultural data is stored and managed.

PROBLEMSTATEMENT

The issue at hand revolves around the outdated and ineffective management of agricultural data, a situation that poses significant obstacles to the growth and sustainabilityoftheagriculturesector.Currently,theprevailingsystemsfallshortintheir capacity to furnish real time data insights to both farmers and stakeholders, thereby resulting in suboptimal decision making and the potential for economic losses. The primary objective of this project is to redefine the problem statement by offering a solution that is not only crystal clear in its operation but also user friendly and highly secureforthemanagementofagriculturaldata.Thisendeavoriscenteredaroundcreating asmart contractthat residesontheEthereumblockchain, atechnologyrenowned for its trustworthiness and transparency. This smart contract will possess the capability to efficiently handle taskssuchasdata addition, querying, and updates.Inessence, it aims to revolutionize the management of agricultural data, addressing existing challenges in the process. Furthermore, it will significantly enhance data privacy and ensure compliance with pertinent regulations, thus heralding a transformative change in the agriculture industry. The ultimate goal is to streamline the entire process of managing agricultural data, rendering it not only easily accessible but also highly reliable and secure. In doing so, it will not only benefit the farming community but also empower stakeholderswithmorerobustdecisionmakingtools,ultimatelyfosteringthegrowthand sustainability of the agriculture sector.

3.   IDEATIONANDPROPOSEDSOLUTION

EMPATHYMAP CANVAS

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

Expresses the need for a more efficient way to manage and access agriculture data.

Wants a secure and user-friendly system for farmers to input and retrieve data.

May request predictive analytics and market insights.

Emphasizes the importance of data accuracy and reliability.

Asks for data privacy and compliance with regulations.

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

Thinks about the potential scalability challenges of blockchain technology.

Expects the system to simplify data management for farmers.

Considers the usability and accessibility of the platform.

Concerned about data security and privacy on the blockchain.

Expects the project to enhance decision-making for farmers.

Farmer
A *farmer* is a person engaged in agriculture, raising living organisms for food or raw materials.

Collaborates with other stakeholders and data contributors.

Engages with the system to input agriculture data and queries.

Relies on the system for important agricultural decisions.

Expects prompt responses and real-time data updates.

May provide feedback or suggestions for improvements.

Excited about the potential of blockchain technology to streamline data management.

Anxious about the security of sensitive agricultural data.

Frustrated with complex and outdated data management systems.

Hopeful that the project will lead to better agricultural outcomes.

Expects trust and transparency in data management.

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?

See an example

IDEATIONANDBRAINSTORMING

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

The problem at hand is the inefficient and fragmented management of agricultural data, hindering informed decision-making and data security for farmers and stakeholders. Existing systems lack user-friendly interfaces and real-time data access, which results in data inaccuracy, inconsistency, and vulnerability. The "Agriculture Docs Chain" project aims to revolutionize agricultural data management, providing a secure and user-friendly solution on the Ethereum blockchain, ensuring data integrity, accessibility, and reliability for better decision-making in the agriculture sector.

2

Brainstorm

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

10 minutes

PARAMASIVAM

- Implement a user-friendly web interface for farmers to interact with the smart contract.
- Integrate oracles to fetch real-world weather and market data for better decision-making.
- Utilize decentralized identity systems to ensure data privacy and security.

ASWIN

- Explore data encryption techniques to protect sensitive information like crop yield and farmer details.
- Investigate interoperability with other blockchains or networks to enhance data sharing.
- Research using NFTs to represent unique farm profiles or certifications.

SRIGANTH

- Ensure regulatory compliance for agriculture data management on the blockchain.
- Focus on scalability and optimize gas costs for transactions.
- Explore partnerships with agricultural organizations to increase adoption.

MUTHUKUMAR

- Develop a mobile app for farmers to access their data and receive notifications.
- Implement an incentive mechanism using a native token for data contributors.
- Consider integrating machine learning algorithms to provide predictive analytics.

MONIESH

- Create a decentralized storage solution for large agricultural datasets.
- Consider the development of a data marketplace for buying and selling agricultural insights.
- Establish a governance model to involve stakeholders in decision-making.
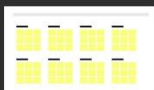
3

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**

USER INTERFACE AND ACCESSIBILITY

Develop a user-friendly web interface and a mobile app for farmers.	Ensure decentralized identity and data privacy.	Explore NFTs for representing unique farm profiles.

DATA SECURITY AND PRIVACY

Implement data encryption and compliance with regulations.	Investigate decentralized storage for large datasets.	Enhance data security with oracles and encryption techniques.

DATA UTILIZATION AND INCENTIVES

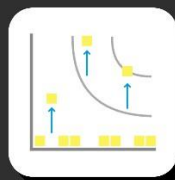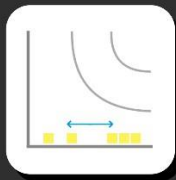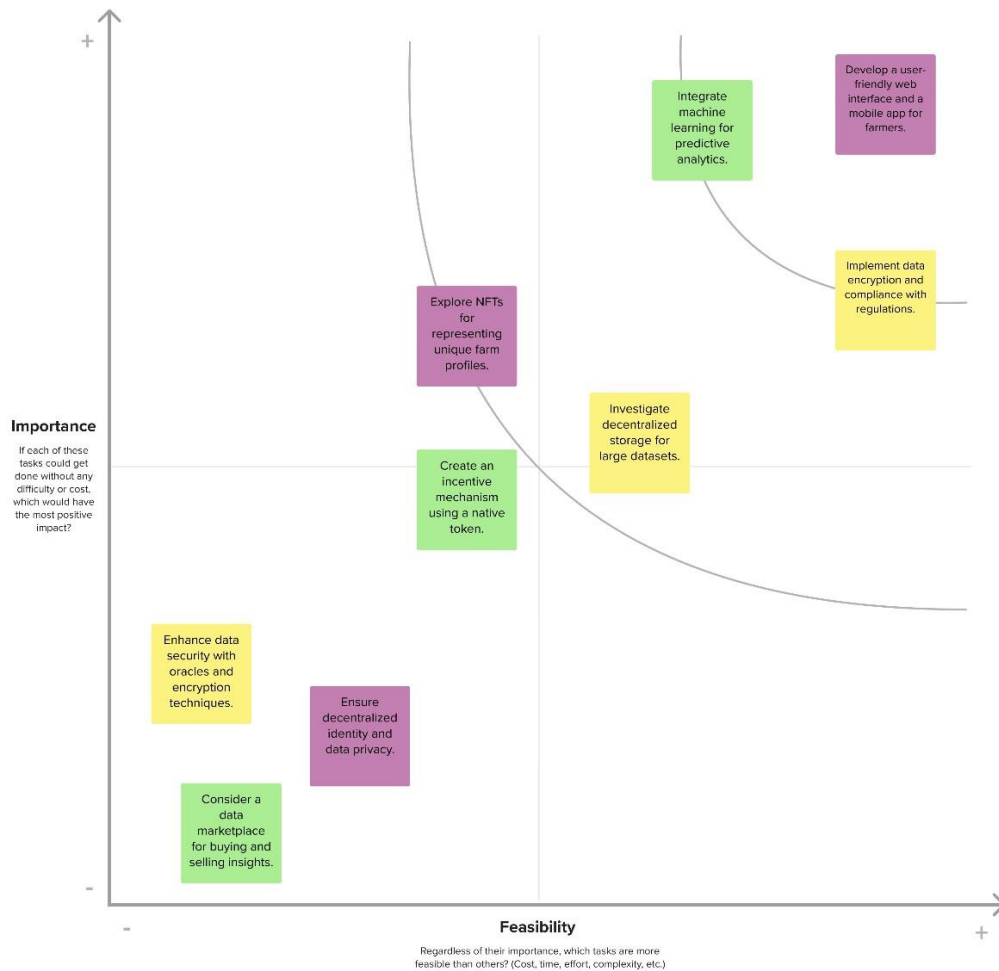|		
Integrate machine learning for predictive analytics.	Create an incentive mechanism using a native token.	Consider a data marketplace for buying and selling insights.
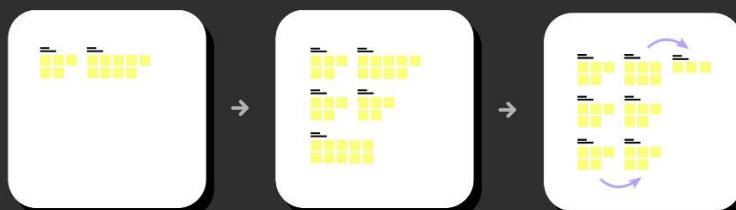
4

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**

Importance

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

Integrate machine learning for predictive analytics.

Develop a user-friendly web interface and a mobile app for farmers.

Implement data encryption and compliance with regulations.

Explore NFTs for representing unique farm profiles.

Investigate decentralized storage for large datasets.

Create an incentive mechanism using a native token.

Enhance data security with oracles and encryption techniques.

Ensure decentralized identity and data privacy.

Consider a data marketplace for buying and selling insights.

Feasibility

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

4.   REQUIREMENTANALYSIS

FUNCTIONALREQUIREMENTS

UserRegistration andAuthentication: Users, primarilyfarmersandstakeholders, must beabletoregisterandauthenticatethemselvessecurely.Thisensuresthat only authorized individuals can access and interact with the system.

User Wallet Integration: The system should allow users to integrate their digital wallets,suchasMetamask,tosecurelysigntransactionsandmanagecryptocurrency within the application.

UserFriendly Frontend: The frontend must feature a straightforward and intuitive user interface with buttons to connect the wallet, add new product entries, update existing data, and retrieve product details. Users should easilynavigate and interact with the platform.

Add Product: Users should have the ability to add agricultural products to the system.ThisincludesinputfieldsforproductID,productname,productdescription, and product quantity. The systemshould validateand storethis data securelyonthe blockchain.

UpdateProduct: Usersmustbeabletoupdateproductinformation,whichinvolves inputfieldsforproductID,productname,productdescription,andproductquantity. This feature ensures that the data remains accurate and uptodate.

Query Product Information: Users should be able to retrieve product details by enteringaspecificproductID.Thesystemwillthenfetchanddisplaythestoreddata, offering quick access to crucial agricultural insights.

Blockchain Smart Contract: The project's core component is the Ethereum block chainbasedsmart contract,writteninSolidity.Itshouldhandletheaddition,querying, and updating of agricultural data securely and transparently.

Data Validation: The system should incorporate data validation mechanisms to ensure that only accurate and relevant information is stored. Invalid or inconsistent data should be rejected.

Security Measures: Robust security measures, including encryption and access controls, must be in place to safeguard sensitive agricultural data. Only authorized users should have access to specific information.

Error Handling: The system should include comprehensive errorhandling procedures to provide informative feedback to users in case of data entry or transaction errors.

Compliance: The solution should adhere to relevant data privacy and regulatory requirements,ensuringthatusers'dataishandledinacompliantandethicalmanner.

Scalability and Performance: Asdata volumesgrow,the systemshould be able to scale efficientlywhile maintaining optimalperformance, ensuring that it canhandle an increasing number of users and data records.

These functional requirements are critical for creating a robust and user friendly agricultural data management system that fulfills the project's objectives of secure, efficient, and accessible data handling on the blockchain.

NONFUNCTIONALREQUIREMENT:

Performance: The system must be responsive and capable of handling a large number of simultaneous users and data transactions efficiently, with minimal latency.

Security: Robust security measures should be implemented to protect against unauthorizedaccess, databreaches, and other securitythreats.Dataencryptionand secure authentication methods are essential.

Reliability: The system must be highly reliable, ensuring that it operates consistently without frequent downtime or data loss.

Scalability: The solutionshould be designed to scale as the volume ofagriculturaldata and the number of users grow over time.

Usability: The user interfaces (web and mobile) should be intuitive, userfriendly, and accessible to individuals with varying levels of technical expertise.

Data Privacy: The system must comply with data privacy regulations and ensure that users' sensitive information is protected.

Auditability: The platform should maintain a comprehensive audit trail, recording all data transactions and interactions with the smart contract.

Disaster Recovery: A robust disaster recovery plan should be in place to ensure datarecovery and system restoration in case of unexpected events or failures.

CostEffectiveness: Theprojectshouldstrivetomaintainacosteffectiveapproach, optimizing gas costs for transactions on the blockchain and other operational expenses.

Interoperability: Thesystemshould bedesigned to interoperatewithotheragricultural datasystems,allowingforseamlessdataexchangeandcollaborationwithexternal stakeholders.

Regulatory Compliance: Ensure that the system complies with agricultural and blockchain related regulations and standards, promoting transparency and trust among users and authority

5.   PROJECTDESIGN

DATAFLOWDIAGRAMANDUSERSTORY
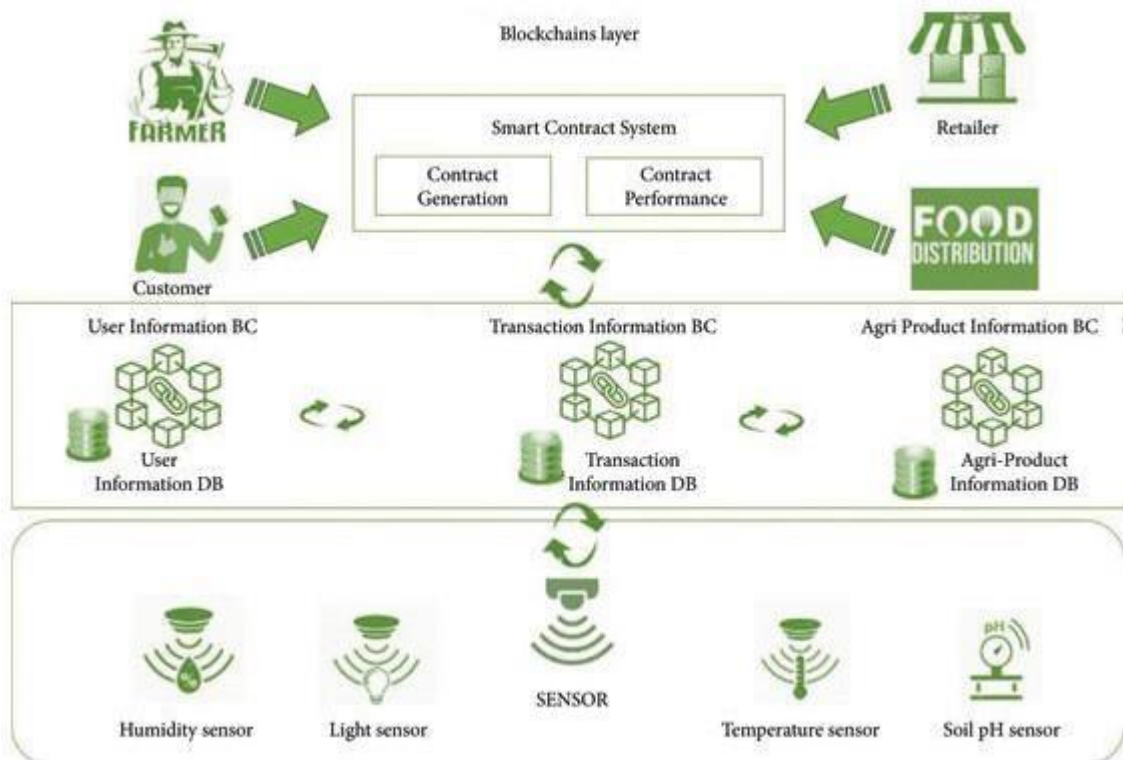
STORY1

As an agriculture data manager, I want to store and manage agricultural product details on the Ethereum blockchain. I need to be able to connect my wallet through Metamask to ensure secure access. Using the frontend interface, I can add new agricultural products by specifying their unique product ID, name, description, and the quantity of the product in the project.Additionally,Ishouldbeabletoupdatethedetailsofexistingproductswhennecessary byprovidingtheproduct'sIDandthenew information. Thisway, Icanmaintainaccurateand up-to-date records of our agricultural projects.

STORY2

As a user of the Agriculture Docs Chain, I want to retrieve product details from the blockchain. To do this, I enter the product's ID into the frontend interface and click the "Get Product Details" button. This feature allows me to access information about specific agricultural products that were previously added to the blockchain. It enables me to quickly andeasilyretrievedataaboutourprojectsandproducts,facilitatinginformeddecision-making and tracking of project progress.

SOLUTIONARCHITECTURE
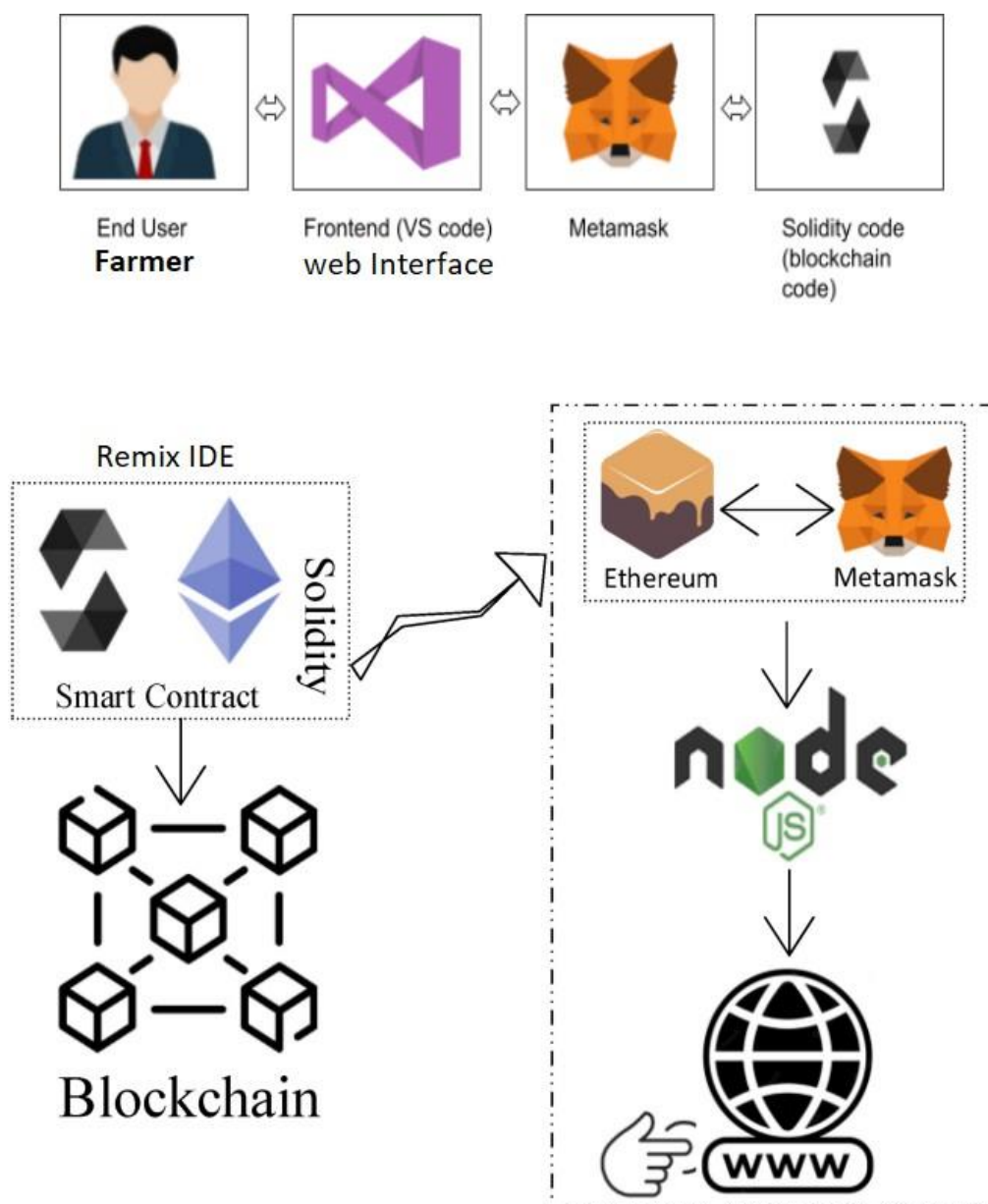
Interaction betweenweb andtheContract

In the diagram, the end user (Farmer) is depicted at the top, using the web interface (Frontend) to access the system.

Metamask serves as the bridge between the user and the Ethereum blockchain, ensuring secure interactions with the smart contract developed in Solidity using the Remix IDE.

The Ethereum module connects to the Ethereum network, while Node.js and www (web servers) facilitate the communication between the frontend and the blockchain.

This architecture ensures data security, accessibility, and reliability, aligning with the project's goals of solving agriculture data management challenges with a robust and user-friendly solution.

6. PROJECT PLANNING & SCHEDULING

TECHNICAL ARCHITECTURE
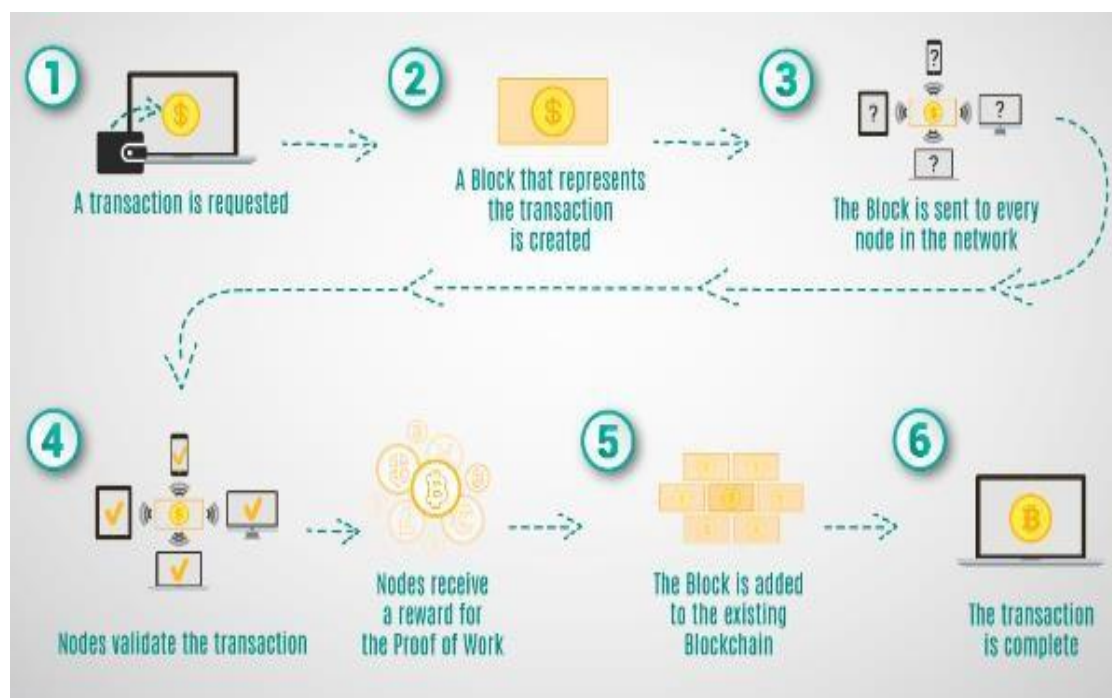
SprintPlanning and Estimation

Sprint planning is a crucial part of our project development process, and it involves selecting and committing to work items from the product backlog to be completed in the upcoming sprint. Here's how we handle it:

• **Reviewing Product Backlog**: Our project team, which includes the product owner, scrum master, and development team, regularly reviews the items in the product backlog. We carefully evaluate user stories and technical tasks, taking into account the evolving needs and priorities of the project.

• **SettingSprintGoals**:Basedontheproduct backlog,ourteamestablishesclearsprint goals. These goals serve as guiding principles for the team's efforts during the sprint, ensuring alignment with the broader project objectives.

• **Breaking Down User Stories**: User stories and tasks are further decomposed into smaller, actionable sub-tasks. This detailed breakdown helps in creating a comprehensive plan for the sprint, ensuring that all necessary steps are considered.

• **Estimating Work**:Our development team employs agile estimation techniques, such as story points and t-shirt sizes, to estimate the effort required for each task. These estimates provide the teamwith a better understanding ofthe scope and complexityofthe work for the sprint.

• **SprintBacklog**:Theselecteduserstoriesandtasks,alongwiththeirestimates,makeupthe sprint backlog. This forms the basis for what the team will work on during the sprint, and it helps in managing and tracking progress effectively.

EstimationTechniques

• **StoryPoints**:Storypointsareusedasarelativemeasureofthecomplexityandeffortneeded to complete a task. Tasks are assigned story point values based on their complexity when compared to reference tasks.

• **T-ShirtSizes**:Toprovideaquickandhigh-levelestimateofeffort,tasksarecategorizedinto t-shirtsizes,suchassmall,medium,andlarge.Thisapproachsimplifiestheestimationprocess, especially for less complex tasks.

SprintDeliveringSchedule

Week1: EstablishtheCore

- Setupthebasicblockchain infrastructure.
- Implement aminimaluserregistrationandauthenticationsystem.
- Developarudimentarytransactionrecordingfeature.
- Focus onfundamentalsecurity measures.

Week2:Expand and Enhance

• Extendtransactionrecordingtosupportmoretransactiontypes.

• Addbasicreal-timeupdatesfortransactionstatus.

• Enhanceuser authenticationwithmulti-factor authentication.

• Begindevelopingauser dashboard.

Week3:Finalizeand Prepare

• Completetheuser dashboardwithadditionalfeatures.

• Performminimalcompliance checks.

• Conduct basictestingandissueresolution.

• Createessentialusersupport resources.

This sprint deliveryschedule outlines the keyobjectives and tasks to be completed ineachof thethreeweeks.Itprovidesaclearroadmapforthedevelopmentteamtofollow,ensuringthat the project progresses smoothly and systematically.

7. CODINGANDSOLUTIONING

FEATURE1

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

contractAgricultureRegistry{ struct
    foodProduct {
        string name;
        stringdescription;
        uint256 quantity;
        address owner;
    }

    mapping(uint256=>foodProduct)publicproducts;
    uint256 public productCount;

    eventProductAdded(uint256productId,stringname,stringdescription,uint256quantity, address
owner);
    eventProductUpdated(uint256productId,stringname,stringdescription,uint256 quantity);
```

```solidity
modifier onlyOwner(uint256 _productId){
    require(products[_productId].owner==msg.sender,"Only the owner can perform this action");
    _;
}

function addProduct(uint256 ProductId, string memory _name, string memory _description, uint256 _quantity) external{

    products[ProductId]=foodProduct(_name,_description,_quantity,msg.sender);
    productCount++;
    emit ProductAdded(productCount,_name,_description,_quantity, msg.sender);
}

function updateProduct(uint256 _productId, string memory _name, string memory _description, uint256 _quantity) external onlyOwner(_productId){
    foodProduct storage product = products[_productId];
    product.name = _name;
    product.description=_description; product.quantity
    = _quantity;
    emit ProductUpdated(_productId,_name,_description,_quantity);
}

function getProductDetails(uint256 _productId) external view returns(string memory name, string memory description, uint256 quantity, address owner) {
    foodProduct memory product=products[_productId];
    return(product.name,product.description,product.quantity,product.owner);
}
}
```

The provided Solidity smart contract, named "AgricultureRegistry," serves as a decentralized registry for tracking and managing food products within the agricultural domain. This contract features a struct named "foodProduct," which encapsulates essential information about each food product, such as its name, description, quantity, and the Ethereum address of the owner.

The contract maintains a mapping, associating each food product with a unique uint256 product identifier. Furthermore, it keeps track of the total product count. The contract includes two significant events, "ProductAdded" and "ProductUpdated," which are triggered to notify users when a new food product is added or when an existing one is updated.

FEATURE 2
FRONTEND(*JAVASCRIPT*)

```
import React,{useState }from "react";
import{Button,Container,Row,Col}from'react-bootstrap'; import
'bootstrap/dist/css/bootstrap.min.css';
import {contract}from "./connector";

functionHome(){
  const[Id, setId]= useState("");
  const[Name,setName]=useState("");
  const [Desc, setDesc] = useState("");
  const [Qty, setQty] = useState("");

  const[Ids,setIds]=useState("");
  const[Names,setNames]=useState("");
  const [Descs, setDescs] = useState("");
  const [Qtys, setQtys] = useState("");
  const [Wallet, setWallet] = useState("");

  const[gId, setGIds]= useState("");
  const[Details,setDetails]= useState("");

  consthandleId=(e)=>{
    setId(e.target.value)
  }
```

```javascript
const handleName=(e)=>{
  setName(e.target.value)
}

const handleDesc=(e)=>{
  setDesc(e.target.value)
}

const handleQty=(e)=>{
  setQty(e.target.value)
}

const handleAddProduct=async()=>{ try
  {
    let tx=await contract.addProduct(Id.toString(),Name,Desc,Qty) let
    wait = await tx.wait()
    alert(wait.transactionHash)
    console.log(wait);
  }catch(error){
    alert(error)
  }
}

const handleIds=(e)=>{
  setIds(e.target.value)
}
```

```javascript
const handleNames = (e) => {
  setNames(e.target.value)
}

const handleDescs = (e) => {
  setDescs(e.target.value)
}

const handleQtys = (e) => {
  setQtys(e.target.value)
}




const handleUpdate = async() => { try {
    let tx = await contract.updateProduct(Ids.toString(), Names, Descs, Qtys) let
    wait = await tx.wait()
    console.log(wait);
    alert(wait.transactionHash)
  } catch(error){
    alert(error)
  }
}




const handleGetIds = async(e) => {
  setGIds(e.target.value)
}

const handleGetDetails = async() => { try {
    let tx = await contract.getProductDetails(gId.toString())
```

```jsx
      let arr = []
      tx.map(e=>{
        arr.push(e)
      })


      console.log(tx);
      setDetails(arr)
    } catch (error) {
      alert(error)
      console.log(error);
    }
  }


  consthandleWallet=async()=>{ if
    (!window.ethereum) {
      returnalert('pleaseinstallmetamask');
    }


    constaddr=awaitwindow.ethereum.request({
      method: 'eth_requestAccounts',
    });


    setWallet(addr[0])


  }


return(
 <div>
  <h1style={{marginTop:"30px",marginBottom:"80px"}}>Agriculture Registry</h1>
    {!Wallet?


      <ButtononClick={handleWallet}style={{marginTop:"30px",marginBottom:"50px"
}}>ConnectWallet</Button>
      :
```

```
      <p style={{ width: "250px",height: "50px",margin: "auto",marginBottom: "50px",
border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)} .. {Wallet.slice(-6)}</p>
    }
  <Container>
   <Row>
    <Col style={{marginRight:"100px"}}>
     <div>
       <inputstyle={{marginTop:"10px",borderRadius:"5px"}}onChange={handleId}
type="number" placeholder="Product Id" value={Id} /><br />
       <input style={{ marginTop:"10px", borderRadius: "5px" }} onChange={handleName}
type="string" placeholder="Product Name" value={Name} /><br />
       <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleDesc}
type="string" placeholder="product description" value={Desc} /><br />
       <inputstyle={{marginTop:"10px",borderRadius:"5px"}}onChange={handleQty}
type="number" placeholder="product quantity" value={Qty} /><br />


       <Button    onClick={handleAddProduct}    style={{   marginTop:   "10px"   }}
variant="primary"> Add Product</Button>
     </div>
     </Col>


        <Colstyle={{marginRight: "100px" }}>
         <div>
          <input    style={{   marginTop:   "10px",   borderRadius:   "5px"   }}
onChange={handleIds} type="number" placeholder="Product Id" value={Ids} /><br />
          <input    style={{   marginTop:   "10px",   borderRadius:   "5px"   }}
onChange={handleNames}type="string"placeholder="ProductName"value={Names}/>
<br />
          <input    style={{   marginTop:   "10px",   borderRadius:   "5px"   }}
onChange={handleDescs}type="string"placeholder="productdescription"value={Descs}
/><br />
          <input    style={{   marginTop:   "10px",   borderRadius:   "5px"   }}
onChange={handleQtys}type="number"placeholder="productquantity"value={Qtys}/>
<br />
```

```jsx
            <Button    onClick={handleUpdate}    style={{    marginTop:    "10px"    }}
variant="primary"> Update Product</Button>
          </div>
        </Col>


  </Row>
  <Row>
      <Col >
        <divstyle={{margin:"auto", marginTop:"100px"}}>
        <input    style={{    marginTop:    "10px",    borderRadius:    "5px"    }}
onChange={handleGetIds} type="number" placeholder="EnterId" value={gId} /><br />


        <Button    onClick={handleGetDetails}    style={{    marginTop:    "10px"    }}
variant="primary">Get Product Details</Button>
        {Details ? Details?.map(e =>{
          return<p>{e.toString()}</p>
        }):<p></p>}
        </div>
      </Col>
  </Row>
  </Container>

 </div>
 )
}


exportdefaultHome;
```

This code is a React component that represents a user interface for an "Agriculture Registry" application. The application seems to interact with a blockchain, likely Ethereum, to perform various actions related to agricultural product registration and retrieval. Here's a brief explanation of what the code provides:

State Management: The component uses the useState hook to manage various states, including productID, name, description, quantity, a list of product details, and the user's wallet address after connecting to a wallet provider like MetaMask.

Adding Products: Users can input product details, including ID, name, description, and quantity. After entering these details and clicking the "Add Product" button, the code attempts to add this product to the blockchain by calling a contract.addProduct function. If successful, it displays a transaction hash.

Updating Products: Users can also update product details by providing a product ID, name, description, and quantity. Clicking the "Update Product" button attempts to update the product's information in a similar manner to adding.

Fetching Product Details: Users can enter a product ID and click the "Get Product Details" button. The code calls a contract.getProductDetails function, likely fetching product details from the blockchain. It then displays the retrieved details in the UI.

Wallet Connection: Before using the application, users are required to connect their Ethereum wallet (e.g., MetaMask) by clicking the "Connect Wallet" button. Once connected, the wallet's address is displayed on the screen.

React Bootstrap: The component uses the React Bootstrap library for styling and UI components like buttons, input fields, and containers.

Overall, this code provides a basic front-end for interacting with a blockchain-based agriculture registry, allowing users to add, update, and retrieve product details while also managing wallet connections.

7.   PERFORMANCETESTING

PERFORMANCEMETRICS

Transaction Throughput: Measure the number of transactions the system can handle per second. High throughput is essential for accommodating a large number of users and data interactions simultaneously.

Latency: Track the time it takes for a transaction to be confirmed and recorded on the blockchain. Lower latency ensures a more responsive system**.**

ErrorRates: Monitorthefrequencyoftransactionfailuresorerrors.Reducingerror rates is crucial for ensuring data accuracy and system reliability.

Scalability: Evaluate how the system scales as the volume of data and the number of users increase. Ensure that the system can handle growth without significant performance degradation.

ResourceUtilization: Assessthesystem'suseofcomputationalresources,memory, and network bandwidth. Efficient resource utilization helps optimize costs and maintain performance.

Gas Costs: Keep track of gas costs associated with Ethereum transactions. Optimizing gas costs is essential for cost-effective operation on the blockchain.

ResponseTime: Measurethetimeittakesforthefrontendtorespondtouseractions.   A lower response time leads to a more satisfying user experience.

Uptime and Availability: Calculate the percentage of time the system is available and operational. High uptime ensures uninterrupted service for users.

Data Retrieval Speed: Evaluate how quickly the system can retrieve and display agricultural data to users. Faster data retrieval contributes to informed decision-making.

Data Storage Efficiency: Analyze how efficiently the system stores data on the blockchain. Efficient data storage minimizes blockchain bloat and optimizes costs.

Security Audits and Vulnerabilities: Regularly conduct security audits and penetration testing to identify and address vulnerabilities, ensuring the safety of user data.

Compliance: Monitor compliance with data privacy and regulatory requirements. Ensure that user data is handled ethically and in accordance with relevant regulations.

User Adoption and Engagement: Track user adoption rates and user engagement with the system. High adoption and engagement indicate the system's value to users.

Feedback and Support Response Time: Measure the time it takes to respond to user feedback and support requests. Quick responses enhance user satisfaction and trust.

Documentation Completeness: Assess the comprehensiveness of documentation for users and developers. Well-documented systems facilitate efficient onboarding and troubleshooting.

By regularly monitoring these performance metrics, the "Agriculture Docs Chain" project can ensure that it operates effectively, efficiently, and securely while meeting user needs and expectations.
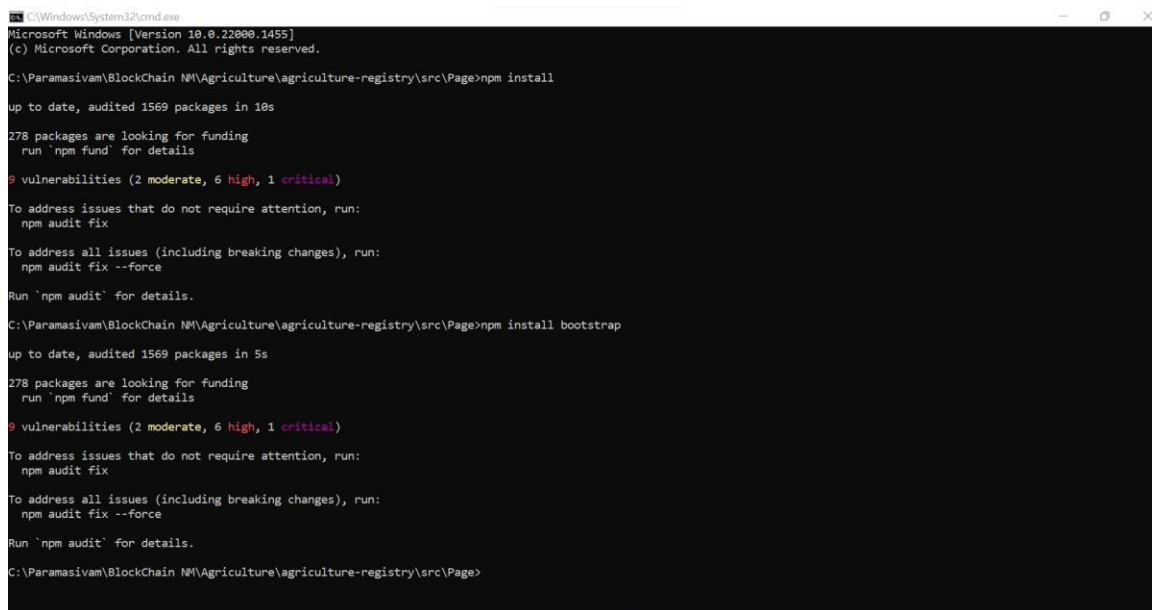
9.RESULTS

OUTPUTSCREENSHOTS

CREATINGASMARTCONTRACT:
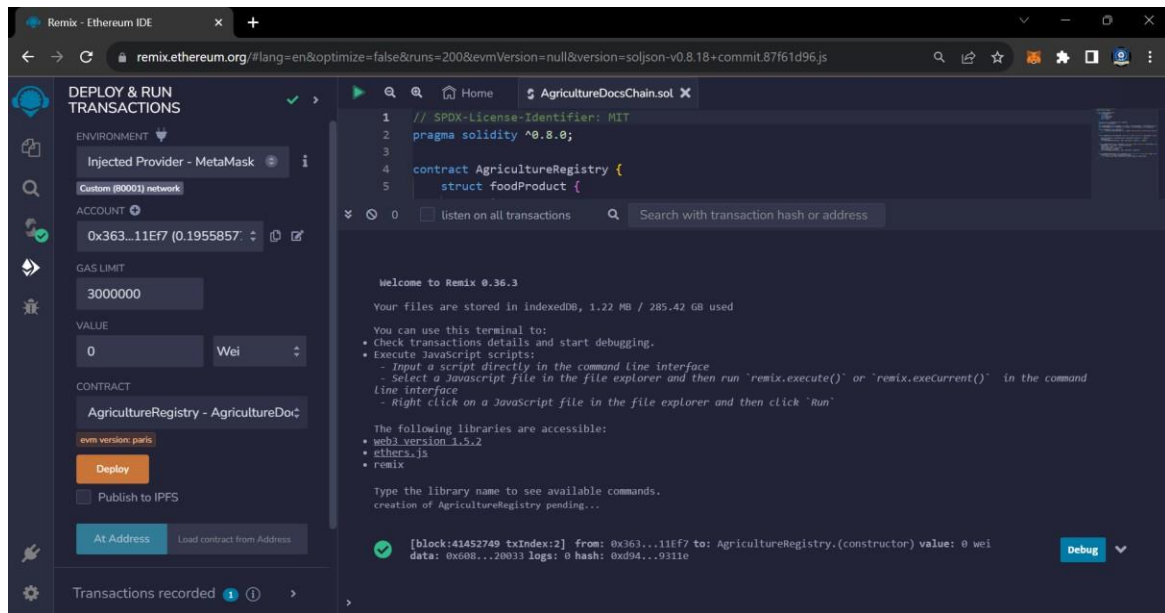
INSTALLINGDEPENDENCY:

HOSTING THE SITE:

OUTPUT SCREENSHOT:

10. ADVANTAGESANDDISADVANTAGES

ADVANTAGES

EfficientDataManagement: Theproject   streamlinesthemanagementofagricultural data, making it more efficient for farmers and stakeholders. It offers a centralized platform for data entry, retrieval, and updates.

Real-Time Data Access: Users gain access to real-time data on crop yields, weather conditions, and market trends, enabling them to make informed decisions quickly.

Security: The use of blockchain technology and robust security measures enhances data security and privacy, reducing the risk of data breaches.

Transparency and Trust: Data stored on the blockchain is transparent and tamper-proof, enhancing trust among users and stakeholders.

User-Friendly Interface: The user interface is intuitive and user-friendly, making it accessible to individuals with varying levels of technical expertise.

Compliance: The system ensures data compliance with relevant regulations and standards, promoting ethical data handling.

Scalability: The system is designed to scale efficiently, accommodating a growing number of users and data records

DISADVANTAGES

Initial Setup Complexity: Implementing the system and configuring smart contractsmaybecomplex,requiringtechnicalexpertiseduringtheinitialsetup.

Gas Costs: Transactions on the Ethereum blockchain come with gas costs, whichcanbeadisadvantageforfrequentandsmalltransactions,affectingcost-effectiveness.

Dependency on Blockchain: The system is reliant on the Ethereum blockchain's performance and network stability, which can be affected by network congestion and scalability issues.

User Training: Users may require training to understand how to use digital wallets and interact with blockchain-based systems.

Data Validation: Ensuring accurate and consistent data input may require additional effort to implement and maintain validation mechanisms.

TechnicalSupport: Technicalissuesand user queries mayrequire responsive technical support, which can be resource-intensive.

RegulatoryChanges: Changesinblockchainandagriculturalregulationsmay require updates and adjustments to the system to remain compliant.

11.   CONCLUSION

In conclusion, the "Agriculture Docs Chain" project represents a pivotal step forward in transforming the landscape of agricultural data management. By leveraging the power of blockchain technology, this project addresses longstanding challenges and inefficiencies in the agriculture sector, offering a range of benefits to both farmers and stakeholders.

The advantages of the system are manifold. It streamlines the traditionally cumbersomeprocessofdata management, providingacentralizedplatformfor adding, querying, and updating agricultural data. Real-time access to critical information,suchascropyieldsandmarkettrends,empowersuserswithtimely insightsformakinginformeddecisions.Theprojectplacesparamountemphasis on securityand transparency, utilizing blockchain to ensure tamper-proof data storage and robust data privacy measures.

However, the project does come with its set of complexities and challenges, including the initial setup intricacies and the issue of gas costs for Ethereum transactions. Yet, with a well-structured technical framework and continuous development efforts, these challenges can be effectively managed.

In the end, the "Agriculture Docs Chain" project has the potential to revolutionize the agriculture industry, offering an efficient, secure, and user-friendly solution for data management. It holds the promise of optimizing productivity, fostering trust, and enhancing data-driven decision-making in agriculture, therebybenefiting farmers, agribusinesses, and stakeholders alike. Asthesystemcontinuesto evolve, it promisesto beabeaconofprogressinthe agriculturesector,driving ittowardsamoreprosperousandsustainable future.

12. FUTURESCOPE

The"AgricultureDocsChain"projecthassignificantfuturescopeandpotential for further development and expansion. Here are detailed future possibilities and areas of growth:

Integration with IoT and Sensors: The project can be extended to integrate withInternet ofThings (IoT)devices and sensorsplaced on farms. This would allow for real-time data collection, such as soil moisture levels, temperature, and humidity. The smart contract could automatically receive and record this data, providing comprehensive insights to farmers.

Advanced Data Analytics: Implementing advanced data analytics and machine learning algorithms can help in the prediction and optimization of agriculturalpractices. Farmerscanreceiverecommendations forcropplanting, irrigation schedules, and pest control based on historical data.

Supply Chain Tracking: Extend the system to track the entire agricultural supply chain, from farmto table. Blockchain can be used to record and verify the origin and quality of products, enhancing food safety and traceability.

Smart ContractsforAgreements: Createsmart contractsfor agreementsand contracts between farmers and buyers. These contracts can automatically execute payments when predefined conditions are met, enhancing trust in business transactions.

Mobile Applications: Develop dedicated mobile applications for a more seamless and accessible user experience, allowing farmers to manage their agricultural data and interact with the blockchain system on the go.

MarketplaceIntegration: Integrateamarketplacewherefarmerscanselltheir agricultural products directly to consumers or businesses. Blockchain can ensure transparent and secure transactions.

Multi-Blockchain Support: Consider supporting multiple blockchains or layer-2 solutions to address scalability issues and provide users with more options for interacting with the system.

Decentralized Identity: Implement decentralized identity solutions to further enhance user privacy and control over their personal information while maintaining compliance with data protection regulations.

Cross-Border Expansion: Explore opportunities to expand the project to support agriculture in different regions or even internationally. This could involve customizing the system to meet specific agricultural needs and regulations in different areas.

Data Monetization: Enable users to monetize their agricultural data by providing it to researchers, agribusinesses, or other stakeholders. This would allow farmers to benefit financially from their data contributions.

Environmental Impact Monitoring: Utilize the system to monitor and track the environmental impact of farming practices, such as water and pesticide usage. This information can be valuable for sustainability efforts and compliance with environmental regulations.

CollaborationwithAgTechCompanies: CollaboratewithAgTechcompanies toenhancetheproject'scapabilitiesandexpanditsreach.Partnershipscanbring in additional resources, expertise, and industry connections.

Astheagricultureindustrycontinuestoevolveandembracedigitalsolutions,the "AgricultureDocsChain"projectholdsthepotentialtoplayapivotalroleinits

transformation. By staying adaptable and responsive to emerging technologies and industry trends, the project can remain at the forefront of agricultural innovation and contribute to the sustainable growth of the sector.

13. APPENDIX

SOURCECODE

agricultureOnBlockchain.sol

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

contractAgricultureRegistry{
    struct foodProduct {
        string name;
        stringdescription;
        uint256 quantity;
        address owner;
    }

    mapping(uint256=>foodProduct)publicproducts;
    uint256 public productCount;

    eventProductAdded(uint256productId,stringname,stringdescription,uint256 quantity,address owner);
    eventProductUpdated(uint256productId,stringname,stringdescription,uint256 quantity);

    modifieronlyOwner(uint256_productId){
        require(products[_productId].owner==msg.sender,"Onlytheownercanperform this action");
        _;
    }
```

```solidity
    functionaddProduct(uint256ProductId,stringmemory_name,stringmemory
_description,uint256_quantity)external{

        products[ProductId]=foodProduct(_name,_description,_quantity,msg.sender);
        productCount++;
        emitProductAdded(productCount,_name,_description,_quantity, msg.sender);
    }


    functionupdateProduct(uint256_productId,stringmemory_name,stringmemory
_description,uint256_quantity)externalonlyOwner(_productId){
        foodProduct storage product = products[_productId];
        product.name = _name;
        product.description=_description; product.quantity
        = _quantity;
        emitProductUpdated(_productId,_name,_description,_quantity);
    }


    functiongetProductDetails(uint256_productId)externalviewreturns(string memory
name, string memory description, uint256 quantity, address owner) {
        foodProductmemoryproduct=products[_productId];
        return(product.name,product.description,product.quantity,product.owner);
    }
}
```

connector.js

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

constabi=[
 {
  "anonymous":false,
  "inputs": [
   {
    "indexed":false,
```

```json
      "internalType":"uint256",
      "name": "productId",
      "type":"uint256"
     },
     {
      "indexed": false,
      "internalType":"string",
      "name": "name",
      "type":"string"
     },
     {
      "indexed": false,
      "internalType":"string",
      "name": "description",
      "type": "string"
     },
     {
      "indexed": false,
      "internalType":"uint256",
      "name": "quantity",
      "type":"uint256"
     },
     {
      "indexed": false,
      "internalType":"address",
      "name": "owner",
      "type":"address"
     }
    ],
    "name":"ProductAdded",
    "type": "event"
   },
   {
    "anonymous":false,
```

```json
    "inputs": [
     {
      "indexed": false,
      "internalType":"uint256",
      "name": "productId",
      "type":"uint256"
     },
     {
      "indexed": false,
      "internalType":"string",
      "name": "name",
      "type":"string"
     },
     {
      "indexed": false,
      "internalType":"string",
      "name": "description",
      "type": "string"
     },
     {
      "indexed": false,
      "internalType":"uint256",
      "name": "quantity",
      "type":"uint256"
     }
    ],
    "name":"ProductUpdated",
    "type": "event"
   },
   {
    "inputs": [
     {
      "internalType":"uint256",
      "name": "ProductId",
```

```json
      "type":"uint256"
     },
     {
      "internalType":"string",
      "name": "_name",
      "type":"string"
     },
     {
      "internalType":"string",
      "name":  "_description",
      "type": "string"
     },
     {
      "internalType":"uint256",
      "name": "_quantity",
      "type":"uint256"
     }
    ],
    "name":"addProduct",
    "outputs": [],
    "stateMutability":"nonpayable",
    "type": "function"
   },
   {
    "inputs": [
     {
      "internalType":"uint256",
      "name": "_productId",
      "type": "uint256"
     }
    ],
    "name":"getProductDetails",
    "outputs": [
     {
```

```json
      "internalType":"string",
      "name": "name",
      "type":"string"
     },
     {
      "internalType":"string",
      "name": "description",
      "type": "string"
     },
     {
      "internalType":"uint256",
      "name": "quantity",
      "type":"uint256"
     },
     {
      "internalType":"address",
      "name": "owner",
      "type":"address"
     }
    ],
    "stateMutability":"view",
    "type": "function"
   },
   {
    "inputs":[],
    "name":"productCount",
    "outputs": [
     {
      "internalType":"uint256",
      "name": "",
      "type":"uint256"
     }
    ],
    "stateMutability":"view",
```

```json
   "type":"function"
 },
 {
  "inputs": [
   {
    "internalType":"uint256",
    "name": "",
    "type":"uint256"
   }
  ],
  "name":"products",
  "outputs": [
   {
    "internalType":"string",
    "name": "name",
    "type":"string"
   },
   {
    "internalType":"string",
    "name": "description",
    "type": "string"
   },
   {
    "internalType":"uint256",
    "name": "quantity",
    "type":"uint256"
   },
   {
    "internalType":"address",
    "name": "owner",
    "type":"address"
   }
  ],
  "stateMutability":"view",
```

```json
      "type":"function"
   },
   {
    "inputs": [
     {
      "internalType":"uint256",
      "name": "_productId",
      "type": "uint256"
     },
     {
      "internalType":"string",
      "name": "_name",
      "type":"string"
     },
     {
      "internalType":"string",
      "name":  "_description",
      "type": "string"
     },
     {
      "internalType":"uint256",
      "name": "_quantity",
      "type":"uint256"
     }
    ],
    "name":"updateProduct",
    "outputs": [],
    "stateMutability":"nonpayable",
    "type": "function"
   }
  ]

  if (!window.ethereum) {
   alert('MetaMaskNotFound')
```

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


exportconstprovider=newethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
exportconstaddress="0x4BAE03dde4D85E8E44F5Bf354C28b5f91735B3B7"


exportconst contract =newethers.Contract(address,abi, signer)
```

Home.js
```
importReact,{useState} from"react";
import {Button,Container,Row,Col} from'react-bootstrap'; import
'bootstrap/dist/css/bootstrap.min.css';
import{contract}from"./connector";


functionHome(){
  const[Id,setId]=useState("");
  const[Name,setName]=useState("");
  const [Desc, setDesc] = useState("");
  const [Qty, setQty] = useState("");


  const[Ids,setIds]=useState("");
  const[Names,setNames]=useState("");
  const [Descs, setDescs] = useState("");
  const [Qtys, setQtys] = useState("");
  const [Wallet, setWallet] = useState("");




  const[gId, setGIds]=useState("");
  const[Details,setDetails]=useState("");
```

```javascript
consthandleId=(e)=>{
  setId(e.target.value)
}


consthandleName=(e)=>{
  setName(e.target.value)
}


consthandleDesc=(e)=>{
  setDesc(e.target.value)
}


consthandleQty=(e)=>{
  setQty(e.target.value)
}




consthandleAddProduct=async()=>{ try
  {
    lettx=awaitcontract.addProduct(Id.toString(),Name,Desc,Qty) let
    wait = await tx.wait()
    alert(wait.transactionHash)
    console.log(wait);
  }catch(error){
    alert(error)
  }
}


consthandleIds=(e)=>{
```

```
      setIds(e.target.value)
  }


  consthandleNames=(e)=>{
    setNames(e.target.value)
  }


  consthandleDescs=(e)=>{
    setDescs(e.target.value)
  }


  consthandleQtys=(e)=>{
    setQtys(e.target.value)
  }




  consthandleUpdate=async()=>{ try
    {
      lettx=awaitcontract.updateProduct(Ids.toString(),Names,Descs,Qtys) let
      wait = await tx.wait()
      console.log(wait);
      alert(wait.transactionHash)
    }catch(error){
      alert(error)
    }
  }




  consthandleGetIds=async(e)=>{
    setGIds(e.target.value)
  }


  consthandleGetDetails=async()=>{
```

```
      try{
        lettx=awaitcontract.getProductDetails(gId.toString())


        let arr = []
        tx.map(e=>{
          arr.push(e)
        })


        console.log(tx);
        setDetails(arr)
      } catch (error) {
        alert(error)
        console.log(error);
      }
    }


  consthandleWallet=async()=>{ if
    (!window.ethereum) {
      returnalert('pleaseinstallmetamask');
    }


    constaddr=awaitwindow.ethereum.request({
      method: 'eth_requestAccounts',
    });


    setWallet(addr[0])


  }


return(
 <div>
  <h1   style={{   marginTop:   "30px",   marginBottom:   "80px"   }}>Agriculture
Registry</h1>
    {!Wallet?
```

```
<ButtononClick={handleWallet} style={{ marginTop: "30px", marginBottom:
"50px" }}>Connect Wallet </Button>
    :
    <pstyle={{width:"250px",height:"50px",margin:"auto",marginBottom: "50px",
border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)} ................{Wallet.slice(-6)}</p>
  }
<Container>
 <Row>
  <Colstyle={{marginRight:"100px"}}>
   <div>
     <input     style={{    marginTop:    "10px",    borderRadius:   "5px"    }}
onChange={handleId}type="number"placeholder="ProductId"value={Id}/><br/>
     <input     style={{    marginTop:    "10px",    borderRadius:   "5px"    }}
onChange={handleName}type="string"placeholder="ProductName"value={Name}
/><br />
     <input     style={{    marginTop:    "10px",    borderRadius:   "5px"    }}
onChange={handleDesc}    type="string"    placeholder="product    description"
value={Desc} /><br />
     <input     style={{    marginTop:    "10px",    borderRadius:   "5px"    }}
onChange={handleQty} type="number"placeholder="productquantity"value={Qty}
/><br />

     <Button   onClick={handleAddProduct}    style={{   marginTop:   "10px"   }}
variant="primary"> Add Product</Button>
   </div>
  </Col>

     <Colstyle={{ marginRight:"100px" }}>
       <div>
         <input    style={{   marginTop:   "10px",   borderRadius:   "5px"   }}
onChange={handleIds}type="number"placeholder="ProductId"value={Ids}/><br
/>
```

```
            <input    style={{    marginTop:    "10px",    borderRadius:    "5px"    }}
onChange={handleNames}        type="string"        placeholder="Product        Name"
value={Names} /><br />
            <input    style={{    marginTop:    "10px",    borderRadius:    "5px"    }}
onChange={handleDescs}        type="string"        placeholder="product        description"
value={Descs} /><br />
            <input    style={{    marginTop:    "10px",    borderRadius:    "5px"    }}
onChange={handleQtys}        type="number"        placeholder="product        quantity"
value={Qtys} /><br />

            <Button    onClick={handleUpdate}    style={{    marginTop:    "10px"    }}
variant="primary"> Update Product</Button>
         </div>
        </Col>

  </Row>
  <Row>
        <Col>
          <divstyle={{margin:"auto",marginTop:"100px"}}>
          <input    style={{    marginTop:    "10px",    borderRadius:    "5px"    }}
onChange={handleGetIds}type="number"placeholder="EnterId"value={gId}/><br
/>

            <ButtononClick={handleGetDetails}style={{marginTop:"10px"}}
variant="primary">Get Product Details</Button>
            {Details?Details?.map(e=>{
              return<p>{e.toString()}</p>
            }):<p></p>}
          </div>
        </Col>
  </Row>
  </Container>

  </div>
```

```
  )
}


exportdefaultHome;
```

App.js

```
import'./App.css';
importHomefrom'./Page/Home'


functionApp(){
  return (
    <divclassName="App">
      <headerclassName="App-header">
       <Home/>
      </header>
    </div>
  );
}


exportdefaultApp;
```

App.css

```
.App{
 text-align:center;
}


.App-logo {
 height:40vmin;
 pointer-events:none;
}


@media(prefers-reduced-motion:no-preference){
 .App-logo {
   animation:App-logo-spininfinite20s linear;
```

```css
  }
}

.App-header{
 background-color:#282c34;
 min-height: 100vh;
 display:flex;
 flex-direction:column;
 align-items: center;
 justify-content: center;
 font-size:calc(10px+2vmin);
 color: white;
}

.App-link {
 color:#61dafb;
}

@keyframesApp-logo-spin{ from
 {
  transform:rotate(0deg);
 }
 to{
  transform:rotate(360deg);
 }
}
```

Index.js
```js
importReactfrom'react';
importReactDOMfrom'react-dom/client';
import './index.css';
importAppfrom'./App';
importreportWebVitals from'./reportWebVitals';
```

```
constroot=ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);


//Ifyouwanttostartmeasuringperformanceinyourapp,passa function
//tologresults(forexample: reportWebVitals(console.log))
//orsendto ananalyticsendpoint.Learnmore:https://bit.ly/CRA-vitals reportWebVitals();
```

Index.css

```
body {
margin: 0;
  font-family:-apple-system,BlinkMacSystemFont,'SegoeUI','Roboto','Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing:antialiased;
  -moz-osx-font-smoothing:grayscale;
}

code{
  font-family:source-code-pro,Menlo,Monaco,Consolas,'CourierNew',
    monospace;
}
```