Project Report

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

Agriculture Docs Chain

1.INTRODUCTION:

Agriculture is a big part of the economy of any country because it helps feed the entire population. It connects and communicates with all of the related industries. If the agriculture base is strong, it is generally regarded as a socially and politically stable society. Many modern farms make use of cutting-edge technology and scientic and technological ideas.

The following are some of the reasons for food supply chain problems and processing environment challenges. The maximization of the prots relies on some farmers' vegetables and fruits with chemicals. Chemical fertilizers, insecticides, and other compounds are used in several plants and fruits.

As a formal denition, the blockchain is a distributed ledger to share transactions or sensitive data across untrusted multiple stockholders in a decentralized network. &e data are recorded in a sequential chain of hash-linked blocks that facilitate the data distribution to be more manageable than other traditional data storage formats. blocks are verified and uploaded into the chain-like system by selected nodes via an agreed consensus protocol. is consensus mechanism allows all the parties to engage in the monitoring process when adding data flow. In addition, the duplicates of these data are stored in all involved nodes to ensure no tampering.

To make agricultural applications more efficient and reliable, we can divide blockchain applications into four categories. The first is the provenance of traceability and food authenticity. The second category is smart agricultural data management. The third category is trading finance in supply chain management. &e last is the category of other information management system.

1.1. Project Overview:

Comparison between Existing Agricultural Schemes and the Proposed Model. The difference between the existing agriculture supply chain (using the centralized database), a standard blockchain-based agricultural supply chain, and the proposed blockchain-based agrarian architecture.

Key Components:

Contract Owner: The contract owner has more control over the system than anyone else. The owner enters the contract into the system and checks to see if the rule gets correctly implemented.

Seed Storage: Seed and other agricultural products are stored in the seed storage.

Supply Shops: They collect and sell a significant quantity of seed, fertilizer, and other agricultural materials to growers.

Producers: The farmers are considered the most basic rung in production. They are in charge of all tasks relating to agricultural planting and harvesting.

Distributors: Distributors are in charge of safely transporting crops from one location to another.

Wholesalers: Wholesalers buy a decent quantity of crops and agriproducts and resell them to retailers.

Retailers: Retailers buy commodities and products from wholesalers and sell them to consumers on a small scale in open markets.

Consumers: They are a large group of individuals that rely on agricultural products. They play a big part in the system by constantly creating demand.

1.2. Purpose:

Physical Layer. A smart contract gets placed in this layer, consisting of numerous sensors, controllers, and IoT devices. These devices are either encapsulated with the smart contract address of the client or discovered by a discovery service. Furthermore, numerous wireless protocols such as Wi-Fi, Zigbee, or LoRa are commonly used in agricultural farms.

Edge Data Layer. The edge nodes make up this layer for deploying containerized microservices, data infrastructure, IoT devices, and QoS control. The edge data layer takes data from the physical layer. It analyzes, compresses, transforms, and splits the data into local and cloud ones. The data rights or the identification of the data creator is initially completed by this layer. As a result, this layer enables offchain verification of tracking data from a cloudbased blockchain. Local servers are in charge of storing off-chain data. This testing overcomes the blockchain implementation problem, privacy, transmission bandwidth, energy usage, and latency, to name a few.

.

Smart Contract Layer. &is layer is responsible for assembling a group of smart contracts. It enables effective, distributed, and heavily automated OASC workflows. Smart law contracts, decentralized autonomous organizations (DAO), and application logic contracts are included in this tier of smart contracts, which go beyond the transfer of simple currency values. &e law contracts specify strict legal remedies to prevent contracting parties from carrying out their obligations. DAO is a blockchainbased community that can design a list of norms expressed in smart contract code. Each participant should respect these rules and have the right to seek recourse if the program gets stopped.

Cloud/Blockchain Layer. This layer combines a cloud storage repository with a blockchain-like ledger that supports three types of blockchains: public, alliance, and private. This layer takes advantage of the alliance chain to include all stakeholders in OASCs. This type of chain combines public and private chains. Cloud/ blockchain layer introduces an InterPlanetary File System (IPFS) and BigchainDB. IPFS is new contentaddressable storage. On each computer, the same file has the same name, and any change in the data file causes the modification in the file name. Due to data storage constraints, the process only keeps the hash value of the file content in the cloud blockchain, and the file itself gets saved at the edge. This layer also utilized BigchainDB, a data storage and search engine, to suit the query specifications of files.

User Layer. The user layer is the main gateway for anyone interested in tracking organic products or maintaining OASCs. The blockchain ecosystem gets linked to this layer via the blockchain bridge, which resembles an Ethereum bridge Metamask and is available as a browser add-on. Furthermore, this layer offers a variety of APIs.

2. LITERATURE SURVEY

2.1. Existing problem:

- The key reason we chose to work with blockchain and incorporate its features into our architecture was the absence of need for third parties.

- Additionally, the control over a decentralized ledger stays with the user rather than a centralized authority.

- Another benefit of blockchain is that there are no data breaches and hacks. However, the scalability of a centralized system with a single server is limited.

- Most blockchain systems implement smart contracts provided by the Ethereum platform and its extension platform, Quorum: they compile using Solidity or Serpent into Ethereum virtual machine (EVM) bytecodes.

- Hyperledger Fabric and Sawtooth, the most active platforms in the Hyperledger family, use Golong, Java, Python, and JavaScript as the major programming languages for smart contract development.

2.2. References:

- G. S. Sajja, K. P. Rane, K. Phasinam, T. Kassanuk, E. Okoronkwo, and P. Prabhue, “Towards applicability of Blockchain in Agriculture sector,” Materials Today Proceedings, vol. 5, 2021.

- J. Lin, Z. Shen, A. Zhang, and Y. Chai, “Blockchain and IoT Based Food Traceability for Smart Agriculture,” in 22 Applied Computational Intelligence and Soft Computing Proceedings of the 3rd International Conference on Crowd Science and Engineering, Singapore, 2018.

- "A New Ecosystem for Educational Credentials: A Progress Report on Blockchain"Author: W. F. Fadel, et al. Published by: The American Academy of Arts and SciencesThis report examines the potential of blockchain technology to create a new ecosystem for educational credentials.

- W. Lin, X. Huang, V. Wang et al., “Blockchain technology in current agricultural systems: from techniques to applications,” IEEE Access, vol. 8, pp. 143920–143937, 2020.

- P. Dutta, T. M. Choi, S. Somani, and R. Butala, “Blockchain technology in supply chain operations: Applications, challenges and research opportunities,” Transportation Research Part E: Logistics and Transportation Review, vol. 142, Article ID 102067, 2020.

- G. Leduc, S. Kubler, and J. P. Georges, “Innovative Blockchain-based farming marketplace and Smart contract performance evaluation,” Journal of Cleaner Production, vol. 306, pp. 1–15, 2021.

2.3. Problem Statement Definition:

2.3.1. Problem statement:

In the face of global climate change, there is an increasing need for a reliable, secure, and transparent system to track, verify, and manage climate-related data and assets. Current methods for monitoring carbon emissions, renewable energy production, or carbon credits trading lack transparency, are often subject to fraud, and have limited cross-border compatibility. To address these challenges, the problem statement is to develop a "agriculture docs chain" system using blockchain technology. This system should enable the secure and decentralized tracking of climate-related activities, assets, and data to ensure accuracy, prevent fraud, and facilitate efficient reporting and trading on a global scale.

Key elements of this problem statement include:

Agriculture Data Tracking: Designing a system that can accurately track agriculture-related data, such as carbon emissions, temperature changes, and renewable energy production, in real-time or near-real-time.

Verification and Transparency: Ensuring that the system provides transparent, immutable records that can be verified by relevant stakeholders, including governments, organizations, and the public.

Security and Fraud Prevention: Implementing robust security measures to prevent fraudulent or unauthorized changes to the data and ensure the integrity of the information.

Interoperability: Creating a system that can function across borders and with different types of climate data, enabling global cooperation and consistency.

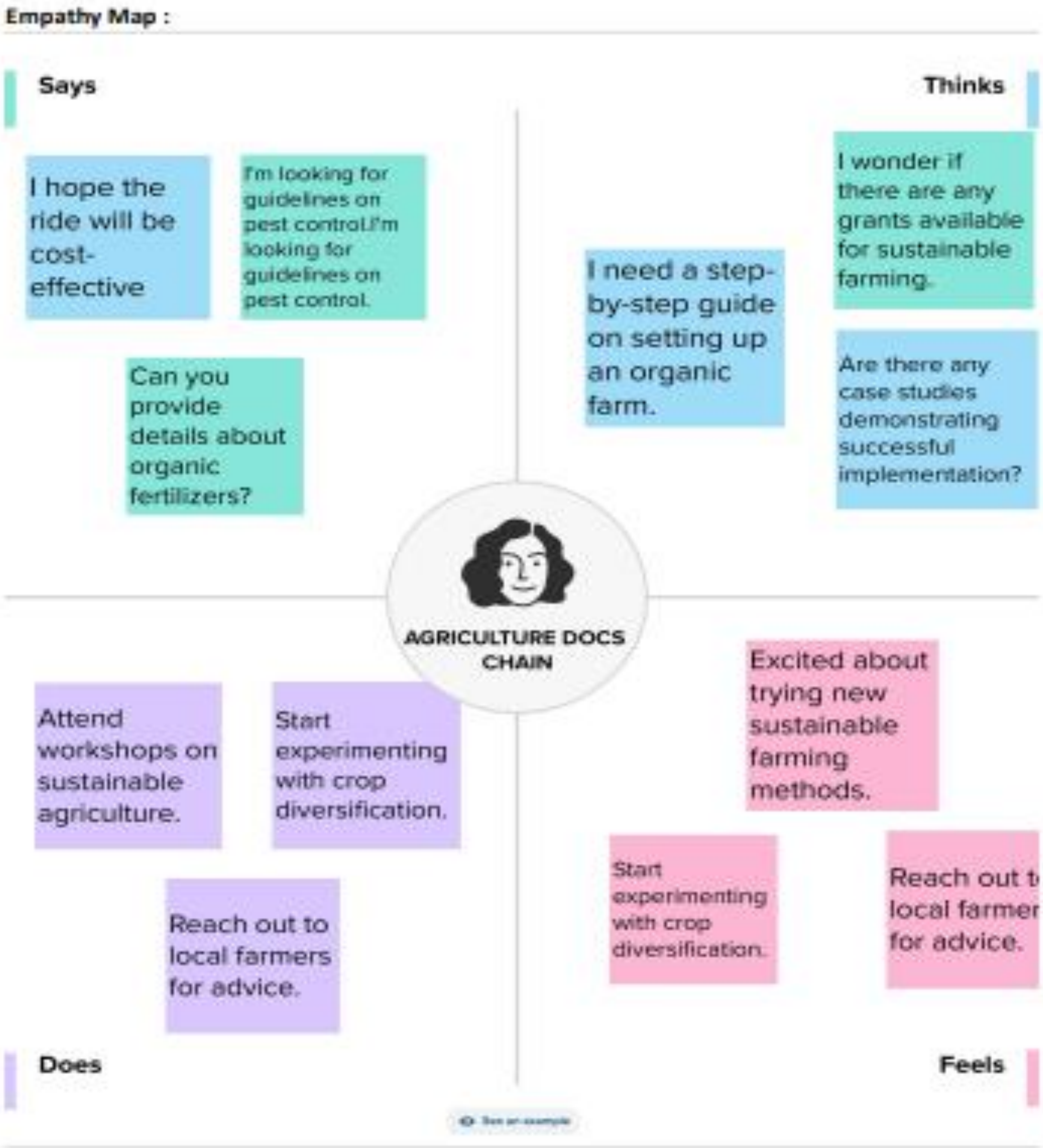
Efficiency and Automation: Developing smart contract functionalities or automation to simplify processes such as carbon credit trading, compliance reporting, and data sharing.

PS	I am (User)	I am Trying to	But	Because	Which Makes me feel
PS-1	User	To Record or Review Transparent Education Data Management	**Data Security and Privacy**:	Conventional systems lack robust security measures.	Implement a permissioned Blockchain network with cryptographic encryption to ensure secure and private data storage and transmission.
PS-2	User	To Record or Review Transparent Education Data Management	**Verification of Academic Credentials**	Manual verification processes are inefficient.	Use Blockchain for transparent, tamper-proof academic credential verification. Institutions can upload data, and verification can be done quickly through a decentralized system.
PS-3	User	To Record or Review Transparent Education Data Management	**Fraudulent Degree Mills**:	Lack of a trusted and centralized authority.	Create a Blockchain-based degree verification system that records and authenticates degrees, making it harder for fraudulent institutions to thrive.

PS-4	User	To Record or Review Transparent Education Data Management	**Inefficient Record Keeping** :	Paper-based and outdated record-keeping methods.	Implement Blockchain to create a single, immutable ledger for student records, ensuring accuracy and making records easily transferable across institutions.
PS-5	User	To Record or Review Transparent Education Data Management	**Credential Ownership and Control**: .	Institutions typically control and manage credentials.	Develop a Blockchain-based solution that allows students to have ownership and control over their academic records, granting them the ability to share them securely as needed.

3. IDEATION & PROPOSED

SOLUTION3.1.Empathy Map



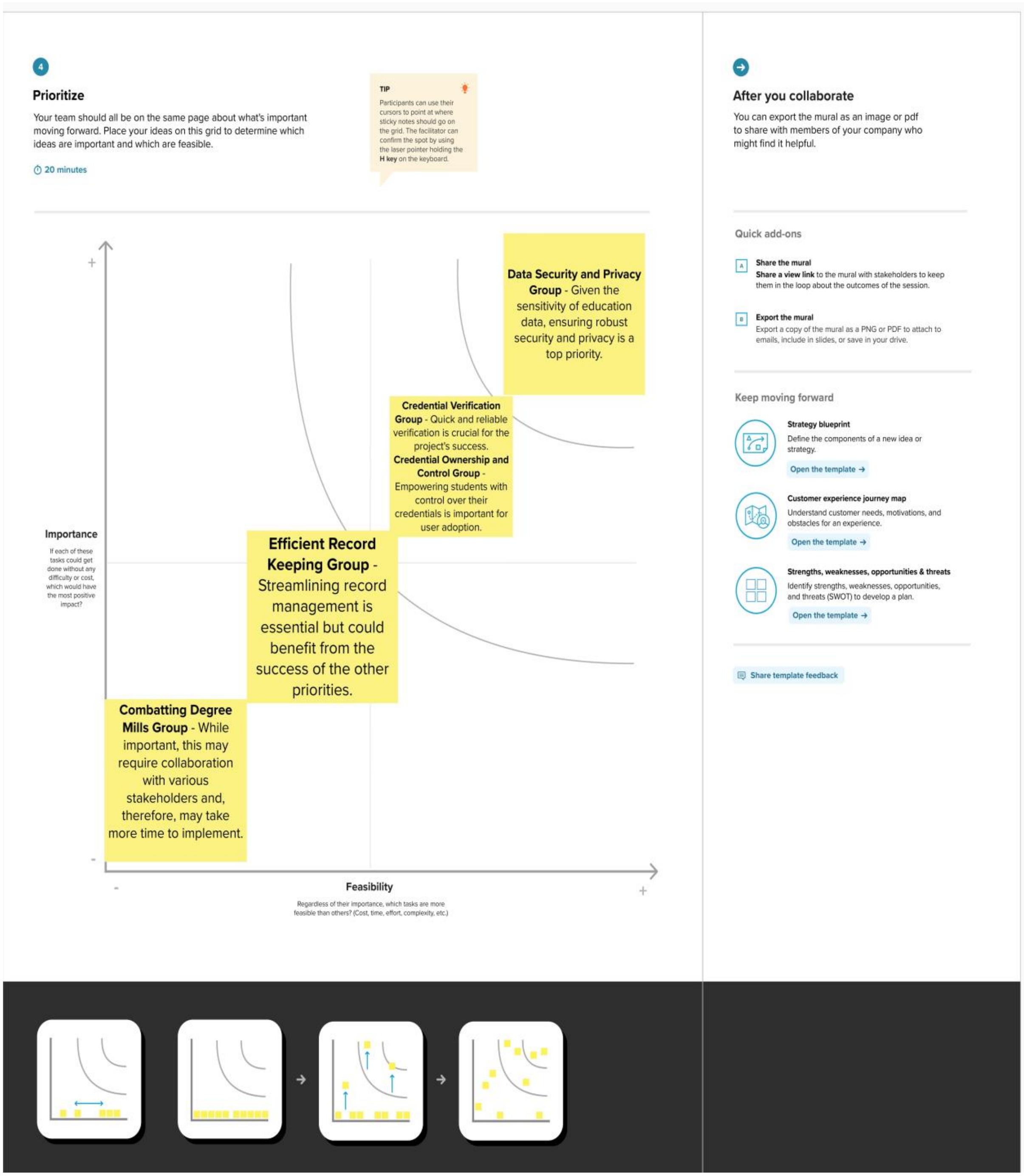
3.2.Ideation & Brainstorming

Brainstorm & Idea Prioritization

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization



4. REQUIREMENT ANALYSIS

4.1. Functional requirement

Following are the functional requirement of the Proposed Solution.

FR. no	Functional Requirement(Epic)	Sub-Requirement
1	User Registration and Authentication	Users (students, institutions, employers, regulators) must be able to register and authenticate their identities securely.
2	Blockchain-Based Record Storage	The system should allow educational institutions to upload and store academic records securely on a blockchain.
3	Credential Verification	Users should be able to verify academic credentials quickly and reliably through the platform.
4	Credential Ownership and Control	Students should have control over who can access and verify their academic records and credentials.

5	Record Transfer and Sharing	Institutions should be able to transfer student records securely to other educational institutions, and students should be able to share their credentials with employers.
6	Anti-Fraud Mechanisms	The system should include mechanisms to detect and prevent fraudulent degree mills from gaining credibility.

4.2. Non-Functional Requirement:

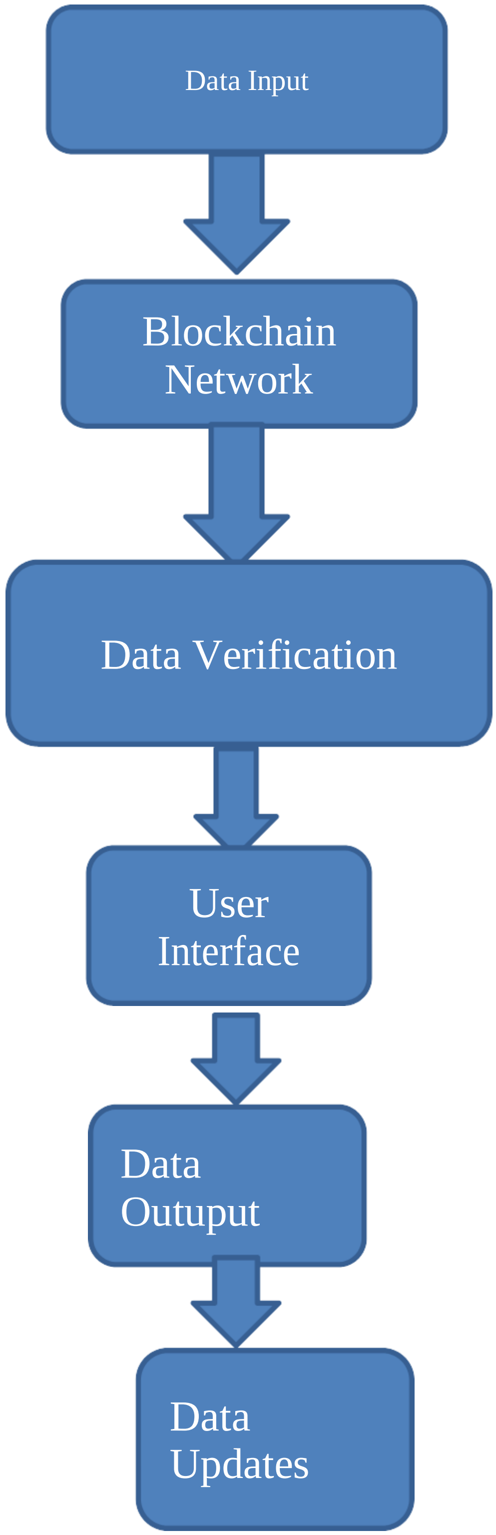
Following are the Non-functional requirement of the Proposed Solution.

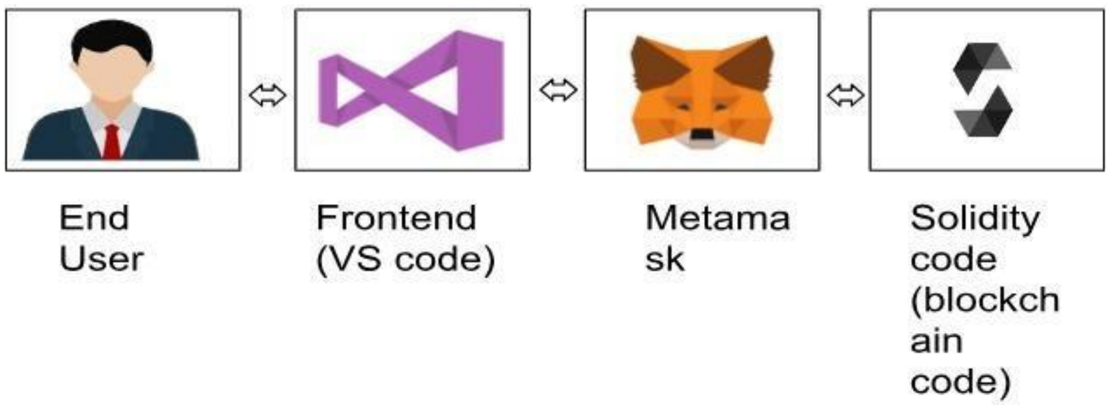
Non-FR. no	Non-Functional Requirement(Epic)	Sub-Requirement
1	Security	The system must ensure the highest level of data security and privacy to protect sensitive educational records and personal information.
2	Scalability	The platform should be scalable to accommodate a growing number of users, educational institutions, and records.
3	Performance	The system must provide fast and reliable academic credential verification and record retrieval.
4	Usability	The user interface should be intuitive, user-friendly, and accessible to individuals with diverse technical backgrounds.
5	Compliance and Regulation	The system must adhere to relevant data protection regulations and ensure transparency in line with educational authorities and regulators' requirements

6	Interoperability	The system should be able to integrate with existing education data systems and provide APIs for third-party applications and services.

5.PROJECT DESIGN

5.1. Data Flow Diagrams & User Stories5.1.1Data Flow Diagrams




End
User
Frontend
(VS code)
Metama
sk
Solidity
code
(blockch
ain
code)

5.1.2.User Stories:

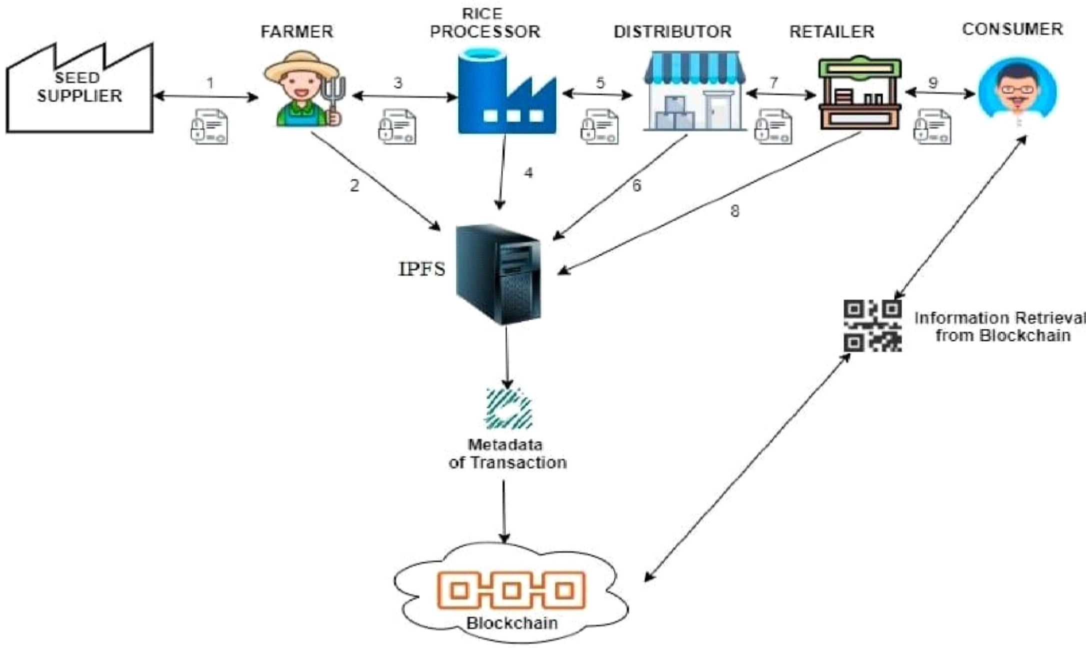
User Story	Acceptance Criteria
Student-As a student, I want to securely upload and manage my academic records on the blockchain.	I can register and authenticate my identity securely. I can upload and view my academic records on the blockchain. I have control over who can access and verify my credentials.
Educational Institution-As an educational institution, I want to upload and store student records on the blockchain.	I can register and authenticate my institution's identity. I can securely upload and store student records on the blockchain. The records are tamper-proof and easily transferable.
Employer-As an employer, I want to quickly verify the academic credentials of potential employees.	I can register and authenticate my identity. I can enter a candidate's details and quickly verify their academic credentials. The verification process is reliable and secure.
Regulatory Authority-As a regulatory authority, I want to ensure compliance and transparency in education data management.	-The system complies with data protection regulations and relevant educational standards. I have access to necessary data for oversight and regulation.
System Administrator-As a system administrator, I want to monitor and maintain the security and performance of the blockchain system.	I can monitor system security and identify potential threats. I can perform regular maintenance and updates. The system's performance is optimized, and issues are addressed promptly.

5.2. Solution Architecture

Solution Architecture is complex process-with may sub -Processes - That Bridges the gap between business problem and Technology solution Its goals are to;

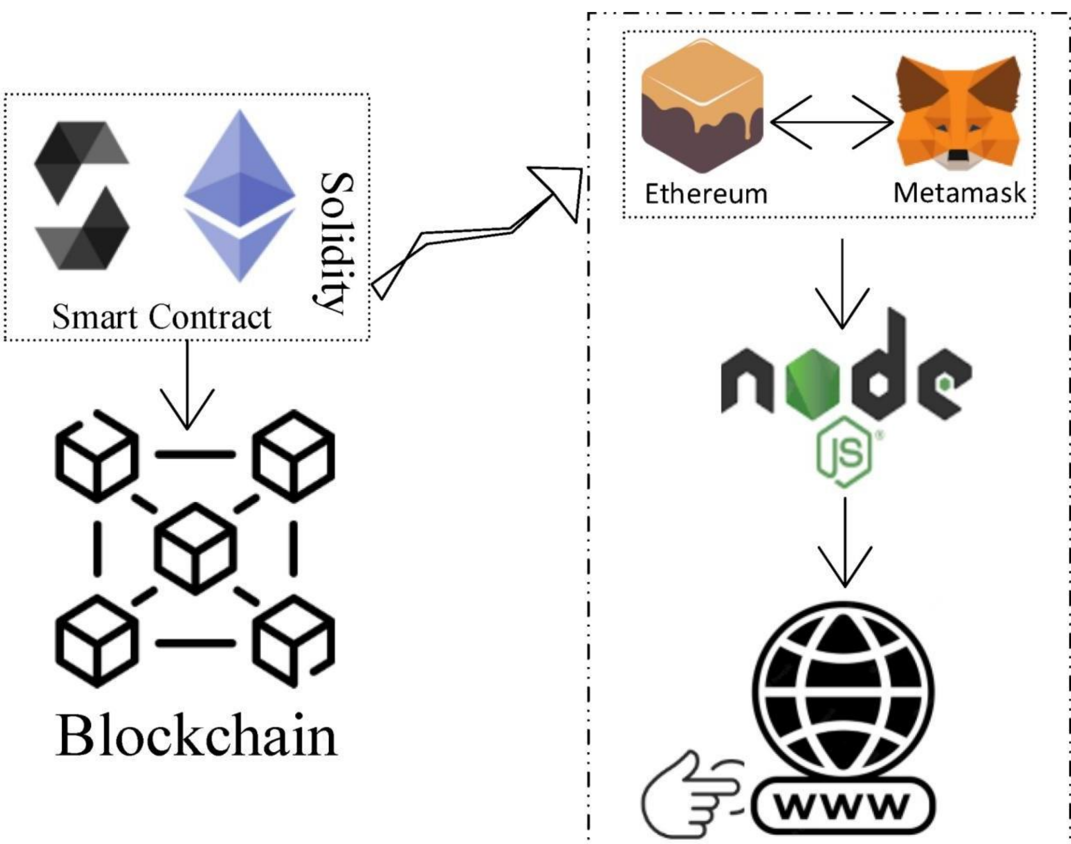
- ❖ Find the best solution to solve existing business problem.
- ❖ Describe the structure,characteristics,behaviour and other aspect of the software to project skectchholder.
- ❖ Define feature,Development phase and Solution requirement.

Provide specification according to which the Solution is defined,managed and delivery.



6.PROJECT PLANNING & SCHEDULING

6.1. Technical Architecture



6.2. Sprint Planning & Estimation

Project Initiation and Requirements Gathering-

Tasks: Define project objectives and scope. Identify stakeholders and their requirements.Create a high-level architecture and design.Estimation:Duration: 2 weeks

User Authentication and Registration

Tasks: Implement user registration and authentication. Set up user profiles.Develop a user-friendly interface for user registration.Estimation:Duration: 3 weeks

Blockchain Integration and Data Storage

Tasks:Implement blockchain integration for secure data storage.Develop data upload and storage features for educational institutions.Estimation: Duration: 4 weeks

Credential Verification and Record Sharing

Tasks: Create a credential verification system for employers and students.Enable record sharing and transfer features.Estimation:Duration: 3 weeks

Security and Compliance

Tasks:Enhance system security with encryption and access controls. Ensure compliance with data protection regulations.Estimation:Duration: 2 weeks

Performance Optimization

Tasks:Optimize system performance for scalability. Conduct performance testing and fine-tuning.Estimation:Duration: 2 weeks

User Acceptance Testing and Feedback

Tasks:Invite users to participate in acceptance testing.Gather feedback and make necessary adjustments.Estimation:Duration: 2 weeks

Documentation and Training

Tasks:Create user documentation and training materials.Estimation:Duration: 1 week

Deployment and Go-Live

Tasks:Prepare for system deployment in a production environment.Estimation:Duration: 2 weeks

Post-Deployment Support and Monitoring

Tasks:Provide ongoing support for users.Monitor system performance and security.Estimation:Duration: Ongoing

6.3. Sprint Delivery Schedule

(2 weeks): Project Initiation and Requirements Gathering Define project objectives and scope.Identify stakeholders and their requirements.Create a high-level architecture and design.

(3 weeks): User Authentication and Registration,Implement user registration and authentication. Set up user profiles.

(4 weeks): Blockchain Integration and Data Storage Implement blockchain integration for secure data storage Develop data upload and storage features for educational institutions.

(3 weeks): Credential Verification and Record Sharing Create a credential verification system for employers and students.Enable record sharing and transfer features.

(2 weeks): Security and Compliance Enhance system security with encryption and access controls.Ensure compliance with data protection regulations.

(2 weeks): Performance Optimization Optimize system performance for scalability.Conduct performance testing and fine-tuning.

(2 weeks): User Acceptance Testing and Feedback Invite users to participate in acceptance testing. Gather feedback and make necessary adjustments.

(1 week): Documentation and TrainingCreate user documentation and training materials.

(2 weeks): Deployment and Go-LivePrepare for system deployment in a production environment.

(Ongoing): Post-Deployment Support and Monitoring Provide ongoing support for users.Monitor system performance and security.

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

7.1. Feature1

Certificate

Revocation

This feature allows the contract owner to revoke a certificate in case of fraud or errors.

In this code, we've added a new function `revokeCertificate` that checks if the certificate exists and whether the sender (contract owner) is the issuer. If these conditions are met, the certificate is revoked by deleting the data. An event `CertificateRevoked` is emitted to record the revocation.

Solidity Code:

```
function revokeCertificate(uint256 certificateId) external onlyOwner {
    require(certificateId <= totalCertificates, "Certificate not found");
    Certificate storage cert = certificates[certificateId];

    // Additional checks for revocation, e.g., fraud detection
    require(msg.sender == cert.issuer, "Only the issuer can revoke the
certificate");

    // Revoke the certificate by clearing the data
    delete certificates[certificateId];

    emit CertificateRevoked(certificateId, cert.studentName,
cert.courseName, now, msg.sender);
}
```

7.2. Feature02
Certificate Lookup by Student Name

This feature allows anyone to look up a certificate by providing the student's name.

In this code, we've added a new function `getCertificateByStudentName`, which iterates through the stored certificates and returns the certificate ID when it finds a match based on the provided student name. If no match is found, it returns 0 to indicate that the certificate was not found.

Solidity Program:

```
function getCertificateByStudentName(string memory studentName)
external view returns (uint256) {
    for (uint256 i = 1; i <= totalCertificates; i++) {
        if (keccak256(abi.encodePacked(certificates[i].studentName)) ==
keccak256(abi.encodePacked(studentName))) {
            return i;
        }
    }
    return 0; // Not found
}
```

7.3. Database Schema (if Applicable)

1. On-Chain Ethereum Data Schema:

In Ethereum, data is structured as a series of smart contract states. Each state represents the storage of data within a smart contract. In the "collegeCertificate" smart contract, the data schema can be described as follows:`owner`: An Ethereum address representing the owner of the smart contract.`totalCertificates`: An unsigned integer that keeps track of the total number of certificates issued.certificates`: A mapping of `uint256` (certificate ID) to `Certificate` struct.The `Certificate` struct itself contains the following fields:`studentName`: A string representing the name of the student.`courseName`: A string representing the name of the course.`DateOfGraduation`: A `uint256` representing the date of graduation. `issueDate`: A `uint256` representing the date when the certificate was issued.`issuer`: An Ethereum address representing the entity that issued the certificate.

This data schema reflects the on-chain data structure used in the Ethereum blockchain for storing certificate information.

2. Off-Chain Ethereum Data:

Off-chain data in an Ethereum-based application typically refers to data that is not stored directly on the blockchain but is referenced or linked to on-chain data. In the context of this smart contract, off-chain data might include additional details about the student, such as contact information or a transcript. This off-chain data would be stored in a traditional database or IPFS (InterPlanetary File System) and referenced in the smart contract using file hashes or other references.

3. Asset Management (if applicable):

The provided smart contract does not directly handle asset management such as digital tokens or fungible/non-fungible assets. If you intend to incorporate asset management into the project, additional smart contracts or standards like ERC-20 (for fungible tokens) or ERC-721 (for non-fungible tokens) would need to be implemented to manage assets on the Ethereum blockchain.

8. PERFORMANCE TESTING

8.1. Performace Metrics

Throughput:

Metric: Transactions per second (TPS)-Description*: Measure the rate at which the system can process certificate issuance and verification transactions. A high TPS is important to handle a large number of requests efficiently.

Latency:

Metric: Response time -Description*: Measure the time it takes for a transaction (e.g., certificate issuance or verification) to be processed and receive a response. Low latency is critical for user satisfaction.

Scalability: - Metric: Scalability under load -Description*: Test how the system performs as the number of users and transactions increases. Ensure that the system can handle growing demands by adding additional nodes or resources.

Resource Utilization:

Metric: CPU and memory usage -Description*: Monitor the utilization of system resources (CPU and memory) during peak loads. Identify potential bottlenecks and ensure efficient resource allocation.

Availability:

Metric: Uptime and downtime-Description: Measure the system's availability over time. Ensure that it is highly available and robust, with minimal downtime.

Stress Testing

Metric: System failure point-Description*: Test the system's limits to identify the point at which it fails or becomes unstable. This helps determine the system's capacity and resilience under extreme conditions.

Security Testing:

Metric: Successful and unsuccessful attacks-Description: Evaluate the system's resistance to common security attacks such as DDoS attacks, data breaches, and unauthorized access.

Load Testing:

Metric: System performance under expected load-Description: Test the system's performance under expected levels of concurrent users and transactions to ensure it meets operational requirements.

Transaction Confirmation Time:

Metric: Time taken to confirm a transaction on the blockchain-Description: Measure how long it takes for a transaction to be confirmed and added to the blockchain. Short confirmation times are important for efficient certificate issuance and verification.

Blockchain Gas Costs:

Metric: Gas costs per transaction-Description*: Calculate the gas costs associated with each transaction. Minimize gas costs to ensure cost-effective operations.

Error Handling and Recovery:

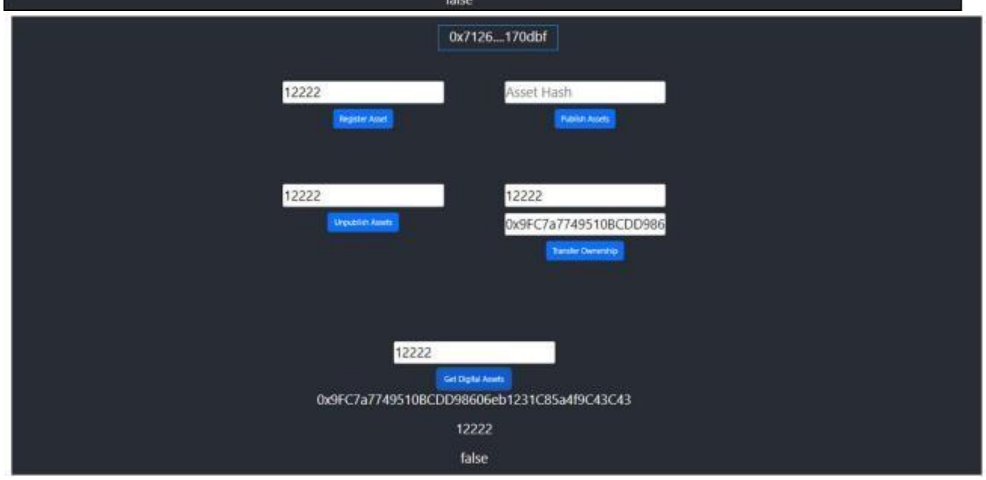
Metric: Error rates and recovery time-Description: Measure the system's ability to handle errors gracefully and recover from failures without data loss or service interruption.

Capacity Planning:

Metric: Resource scalability-Description: Plan for future capacity needs based on performance metrics and expected growth in users and data.

Result:

9.1.Ouput Screenshot:



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Reduction of product losses in transportation and storage.
- Increasing of sales.
- Dissemination of technology, advanced techniques, capital and knowledge among the chain partners.
- Better information about the flow of products, markets and technologies.
- Transparency of the supply chain.
- Tracking & tracing to the source.
- Better control of product safety and quality.
- Large investments and risks are shared among partners in the chain.

DISADVANTAGES:

- Complexity: Implementing blockchain technology can be complex and require expertise. Setting up and maintaining the system might be challenging for some educational institutions.

- Scalability:Scalability can be an issue, especially on public blockchains, as the system needs to accommodate a large number of transactions and users, which can lead to increased costs.

- Regulatory Compliance:Adhering to data protection regulations and ensuring that the system complies with various regional and international standards can be a significant challenge.

- Access and Inclusion: Some students might not have access to the technology required to interact with blockchain-based systems, potentially excluding certain demographics.

- Initial Implementation Costs: The initial setup and integration of blockchain technology may require a significant investment of time and resources.

- Data Recovery: In the event of data loss, recovery from a blockchain can be challenging, and data may be permanently lost.

- User Adoption: Users, including educational institutions and employers, may be resistant to change or unfamiliar with blockchain technology, which could hinder adoption.

- Maintenance: Continuous maintenance, security monitoring, and updates are essential to keep the system running efficiently, which can be resource-intensive.

11.CONCLUSION:

- In conclusion, blockchain and IoT technologies can aid in developing a secure, transparent, open, and innovative ecological agriculture system that involves all participants.

- The is work aims to provide a possible technique to build practical blockchain-based applications and change the agriculture industry, even though the evolution of blockchain and agriculture research studies is still in its infancy.

- The is model is considered a prototype for reducing financial loss and agricultural pollution.

11. FUTURE SCOPE:

The "Transparent Education Data Management Using Blockchain" project has the potential for significant future developments and enhancements. Some of the future scope areas for this project include:

1. Interoperability: The project can expand to enable interoperability with other educational and professional networks. This would allow for seamless data transfer between various institutions and organizations, facilitating a broader exchange of academic credentials.

2. Integration with Learning Platforms: Integrating the blockchain system with existing learning management platforms can streamline the record-keeping process, making it easier for educators to update records and students to access their academic history.

3.Smart Contracts for Verification:Smart contracts can be used to automate and enhance the verification process further. Employers, educational institutions, and regulators can rely on self-executing smart contracts for instant, secure credential validation.

4.Global Credential Verification Standardization:Collaboration with international bodies and governments to establish a standardized protocol for global credential verification can provide greater credibility and acceptance of blockchain-based academic records worldwide.

5.Blockchain-based Diplomas and Certificates:Instead of merely digitizing traditional paper-based certificates, the project can explore creating blockchain-native diplomas and certificates, which are verifiable and secure by design.

6. User-Friendly Mobile Applications:Developing user-friendly mobile applications for students, educational institutions, and employers can promote widespread adoption and ease of use.

7. Extended Ecosystem: The project can extend its ecosystem to cater to other education-related services, such as scholarship and financial aid management, and academic transcripts.

8. Privacy Enhancements: To address privacy concerns, the system can explore more advanced privacy-preserving technologies, allowing individuals to share selected portions of their records without revealing the entire document.

9.Data Recovery Mechanisms: The development of efficient data recovery mechanisms in case of accidental data loss can provide a safety net for the permanent storage of academic records on the blockchain.

10.Blockchain Scalability Solutions: As blockchain scalability is a concern, the project can stay up to date with blockchain advancements and implement scaling solutions as they evolve.

12. APPENDIX Source Code

SOLIDITY CODE:

```
Solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract AgricultureDocumentChain {
    address public owner;
    uint public documentCount = 0;
    struct Document {
        uint id;
        string title;
        string content;
        address owner;
    }

        mapping(uint => Document)
         public documents;
                event
         DocumentCreated(uint id,
         string title,
        address owner);
                constructor() {
                    owner =
         msg.sender;
                }
                modifier onlyOwner()
         {

         require(msg.sender ==
         owner, "Only the owner
        can perform this action");
                    _;
                }
```

```
function createDocument(string memory title,
string memory content) public {
            documentCount++;
documents[documentCount] = Document(documentCount, title, content,
msg.sender);
           emit DocumentCreated(documentCount, title, msg.sender);
       }
}

           totalCertificates = certificateId;

           emit CertificateIssued(
               certificateId,
               studentName,
               courseName,
               issueDate,
               msg.sender
           );
        }

        function getCertificate(
           uint256 certificateId
        ) external view returns (string memory, string memory, uint256,
    uint256, address) {
           Certificate memory cert = certificates[certificateId];
           return (cert.studentName, cert.courseName,
    cert.DateOfGraduation, cert.issueDate, cert.issuer);
        }
    }
```

Java code:

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

const abi = [
 {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "uint256",
    "name": "certificateId",
    "type": "uint256"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "studentName",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "courseName",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "uint256",
    "name": "issueDate",
    "type": "uint256"
   },
```

```
   {
   "indexed": true,
   "internalType": "address",
   "name": "issuer",
   "type": "address"
  }
 ],
 "name": "CertificateIssued",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "string",
   "name": "studentName",
   "type": "string"
  },
  {
   "internalType": "string",
   "name":  "courseName",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "_dateOfGraduation",
   "type": "uint256"
  },
  {
   "internalType": "uint256",
   "name": "issueDate",
   "type": "uint256"
  }
 ],
 "name": "issueCertificate",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
},
```

```json
{
"inputs": [
 {
  "internalType": "uint256",
  "name": "",
  "type": "uint256"
 }
],
"name": "certificates",
"outputs": [
 {
  "internalType": "string",
  "name": "studentName",
  "type": "string"
 },
 {
  "internalType": "string",
  "name":  "courseName",
  "type": "string"
 },
 {
  "internalType": "uint256",
  "name": "DateOfGraduation",
  "type": "uint256"
 },
 {
  "internalType": "uint256",
  "name": "issueDate",
  "type": "uint256"
 },
```

```
    {
     "internalType": "address",
     "name": "issuer",
     "type": "address"
    }
   ],
   "stateMutability": "view",
   "type": "function"
  },
  {
   "inputs": [
    {
     "internalType": "uint256",
     "name": "certificateId",
     "type": "uint256"
    }
   ],
   "name": "getCertificate",
   "outputs": [
    {
     "internalType": "string",
     "name": "",
     "type": "string"
    },
    {
     "internalType": "string",
     "name": "",
     "type": "string"
    },
    {
     "internalType": "uint256",
     "name": "",
     "type": "uint256"
    },
    {
     "internalType": "uint256",
     "name": "",
     "type": "uint256"
    },
    {
     "internalType": "address",
     "name": "",
     "type": "address"
    }
   ],
```

```
{
 "inputs": [],
 "name": "owner",
 "outputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [],
 "name": "totalCertificates",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "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 =
"0x76118a37cCbf2b99Cc371F9E1B5017065103d5c1"
```

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

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon"
href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is
installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the
build.
      Only files inside the `public` folder can be referenced from the
HTML.
```

```
      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico"
will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

Github:

https://github.com/77lokesh/Agriculture-Docs-Chain-

Project Video Demo Link:

https://drive.google.com/file/d/1EAd3PmS7q_7Od_6YgZPZfvZ4RL5p1teD/view?usp=sharing