

BLOCKCHAIN – BASED AGRICULTURE DATA MANAGEMENT SYSTEM

NAAN MUDHALVAN

(2023 – 2024)

PROJECT REPORT

Submitted By

TEAM ID : NM2023TMID06191

SUTHARSAN T (950520106020)

JENIL JEBA EPHRAIM J (950520106008)

MANOJ THANGA BALAJI S (950520106012)

NISHANTH K (950520106013)

SIVA ARUN KUMAR S (950520106017)

In Partial Fulfilment for the Award of the Degree

Of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Dr . SIVANTHI ADITANAR COLLEGE OF ENGINEERING,

TIRUCHENDUR - 628 215.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	3
	1.1 Project Overview	3
	1.2 Purpose	3
2	LITERATURE SURVEY	4
	2.1 Existing problem	4
	2.2 References	4
	2.3 Problem Statement Definition	5
3	IDEATION & PROPOSED SOLUTION	6
	3.1 Empathy Map Canvas	6
	3.2 Ideation & Brainstorming	7
4	REQUIREMENT ANALYSIS	10
	4.1 Functional requirement	10
	4.2 Non-Functional requirements	11
5	PROJECT DESIGN	13
	5.1 Data Flow Diagrams & User Stories	13
	5.2 Solution Architecture	15
6	PROJECT PLANNING & SCHEDULING	16

	6.1 Technical Architecture	16
	6.2 Sprint Planning & Estimation	16
	6.3 Sprint Delivery Schedule	18
7	CODING & SOLUTIONS	19
	7.1 Feature 1	22
	7.2 Feature 2	23
	7.3 Database Schema	23
8	PERFORMANCE TESTING	24
	8.1 Performance Metrics	24
9	RESULTS	25
	9.1 Output Screenshots	25
10	ADVANTAGES & DISADVANTAGES	28
11	CONCLUSION	29
12	FUTURE SCOPE	30
13	APPENDIX	31
	Source Code	31
	GitHub & Project Demo Link	50

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW:

In an era where health and well-being are paramount, understanding and managing our dietary habits have never been more critical. The modern food landscape offers a vast array of options, from tantalizing indulgences to wholesome nourishment. Amid this abundance, it's easy to lose sight of the nutritional content of what we consume. This is where the "Blockchain-Enhanced Agriculture Data Management System" project steps in as a game-changing solution.

Agriculture is the backbone of our society, providing sustenance and livelihoods to billions around the globe. However, the management of agriculture data has often been complex and lacking in transparency. With the rise of blockchain technology, we have an opportunity to revolutionize the way agriculture data is stored, accessed, and updated. The "Blockchain-Enhanced Agriculture Data Management System" aims to leverage blockchain's immutable and transparent nature to provide a secure and efficient platform for recording, querying, and updating critical agriculture data.

In an age when the agricultural industry faces numerous challenges, including the need for traceability, data security, and accessibility, a blockchain-enhanced system is no longer a luxury but a necessity. This system serves as a digital ledger for agriculture data, offering invaluable insights into the cultivation and management of agricultural products. Whether you are a farmer seeking a reliable record-keeping solution, a distributor in need of a transparent supply chain, or a consumer interested in

the source of your food, the blockchain-enhanced agriculture data management system becomes a trustworthy partner.

This project aims to create a robust smart contract on the Ethereum blockchain, enabling users to record agriculture data, query information related to crops, livestock, and supply chains, and update records as needed. It will enhance data privacy and security while promoting transparency and traceability in the agricultural sector.

1.2 PURPOSE:

The primary purpose of the "Blockchain – Based Agriculture Data Management System" is to empower individuals, farmers, stakeholders, and consumers to monitor and manage critical agriculture data efficiently. This system is designed to address the following key objectives:

Data Transparency: The system aims to enhance the transparency of agriculture data by utilizing blockchain technology. It allows farmers to record data related to their crops and livestock, creating an immutable and easily accessible ledger. This transparency benefits consumers, distributors, and other stakeholders by providing insights into the source and journey of agricultural products.

Data Security: Ensuring data security is a paramount concern in the agriculture industry. The blockchain-enhanced system incorporates robust security measures to protect data from unauthorized access, tampering, or data breaches. Users can trust that their information is stored securely.

Data Accessibility: The system facilitates easy and secure access to agriculture data for authorized users. Farmers, distributors, and consumers can query the blockchain for specific information, promoting informed decision-making and traceability throughout the supply chain.

Data Efficiency: The project seeks to streamline data management processes in agriculture. By providing a digital platform for data recording and retrieval, it reduces the administrative burden and potential errors associated with traditional record-keeping methods.

Sustainability: Agriculture data is crucial for addressing global food security and sustainability challenges. This system can contribute to more sustainable agricultural practices by facilitating data-driven decisions and improving resource management.

The "Blockchain – Based Agriculture Data Management System" is not just a technological innovation; it's a solution that aligns with the needs of the modern agricultural landscape. It puts the power of data management into the hands of those who cultivate, distribute, and consume agricultural products, ultimately leading to more informed, secure, and sustainable practices in the industry.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM:

In the realm of food tracking systems, several challenges and issues have been identified in the existing solutions. These issues necessitate the development of a more reliable and user-friendly system for monitoring and managing dietary habits and nutritional intake. The following problems are prevalent in current food tracking systems:

Manual Data Entry: Users often rely on manual data input, which can lead to inaccuracies. Estimating portion sizes and ingredients is an imprecise process, and users may not always be honest or consistent in their reporting.

Limited Food Databases: Many food tracking apps depend on preloaded food databases, which may not encompass all regional or culturally diverse foods. This limitation means users might struggle to find accurate entries for homemade or less common dishes.

Data Reliability: The reliability and quality of data in these databases can vary. Some entries may be incomplete, incorrect, or outdated, leading to misinformation regarding nutritional content.

Barcode Scanning Limitations: While some apps allow users to scan barcodes for packaged foods, not all products have scannable barcodes. Additionally, the accuracy of barcode recognition can fluctuate, which can affect the system's effectiveness.

User Engagement: Many users start using food tracking apps but lose interest or motivation over time. Sustaining user engagement and

encouraging long-term use is a common challenge faced by these applications.

Privacy Concerns: Collecting and storing personal dietary information can raise privacy concerns. Users may be hesitant to share detailed food consumption data, which can limit the effectiveness of the system and discourage certain individuals from using it.

These existing problems in food tracking systems highlight the need for a more robust and user-centric solution. The "Blockchain – Based Agriculture Data Management System" seeks to address these challenges by offering a secure, transparent, and efficient platform for agriculture data management, catering to the unique requirements of the agricultural industry.

2.2 REFERENCES:

1. Adithya Vadapalli, Swapna Peravali, Venkata Rao Dadi, "Smart Agriculture System using IoT Technology", International Journal of Advance Research in Science and Engineering, Vol.9, pp.58-65, Sep.2020.
2. Anirban Kumar, Heshalini Rajagopal, "Automated Seeding and Irrigation System using Arduino", Journal of Robotics, Networking and Artificial Life, Vol. 8(4), pp. 259–262, March 2022.
3. Bhanu K.N., Mahadevaswamy H.S., Jasmine H.J., "IoT based Smart System for Enhanced Irrigation in Agriculture", International Conference on Electronics and Sustainable Communication Systems, pp.760-765, Aug. 2020.
4. Bhuwan Kashyap, Ratnesh Kumar, "Sensing Methodologies in Agriculture for Soil Moisture and Nutrient Monitoring", IEEE Access, Vol. 9, pp.14095-14121, Jan. 2021.

5. Fidaus Kamaruddin, Nik Noordini Nik Abd Malik, Noor Asniza Murad, Nurul Mu'azzah Abdul Latiff, Sharifah Kamilah Syed Yusof, Shipun Anuar Hamzah, "IoT-based intelligent irrigation management and monitoring system using Arduino", TELKOMNIKA, Vol.17, No.5, pp.2378~2388, October 2019.

2.3 PROBLEM STATEMENT DEFINITION:

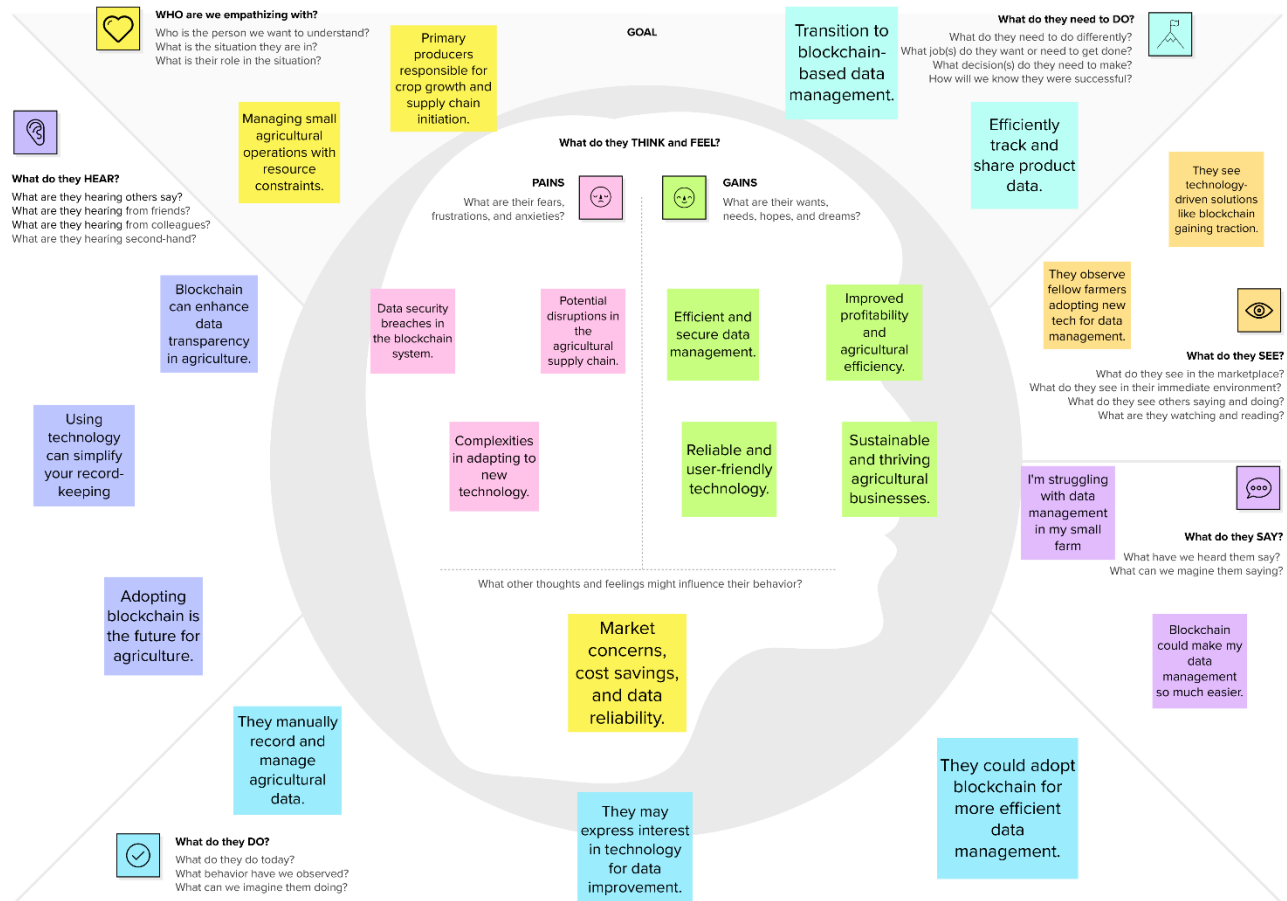
In the agriculture sector, the absence of an efficient and user-friendly data management system hinders stakeholders from effectively monitoring and managing agricultural data, leading to challenges in ensuring the security, transparency, and efficiency of the supply chain. Existing agricultural data management systems often suffer from issues related to data accuracy, accessibility, and transparency, which fail to provide a comprehensive and satisfying solution for users seeking to make informed decisions in agriculture.

This project aims to develop a robust and user-centric agriculture data management system that overcomes these challenges and empowers users to easily and accurately record, access, and update agriculture data, enhancing the transparency and traceability of the entire supply chain.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

In the development of the "Blockchain – Based Agriculture Data Management System," the project team engaged in a comprehensive ideation and brainstorming process to explore various potential solutions and features that can enhance the system's functionality. The following innovative ideas and strategies were considered:

- **QR Code Labelling:**

An implementation of QR codes on agricultural products' packaging that allows consumers to scan and access comprehensive information about the product's origin, production methods, and safety checks. This technology

promotes transparency in the supply chain and ensures that consumers can make informed choices about the products they purchase.

- **IoT Sensors:**

Leveraging the power of the Internet of Things (IoT) sensors to monitor environmental conditions such as temperature, humidity, and other factors during the transportation and storage of agricultural products. This data is invaluable for maintaining food quality and safety, as it enables real-time tracking and intervention in case of adverse conditions.

- **Mobile Apps for Consumers:**

The development of user-friendly mobile applications designed for consumers. These apps enable users to scan product labels, access allergen information, and receive notifications regarding product recalls or safety alerts. They empower consumers with information that enhances their decision-making when selecting agricultural products.

- **Supplier Verification:**

Establishing a rigorous system for supplier verification and certification, ensuring that suppliers meet specific standards and compliance requirements before their products can enter the supply chain. This measure guarantees that agricultural products meet predefined quality and safety standards from the source.

- **Predictive Analytics:**

Utilizing predictive analytics to forecast and mitigate supply chain disruptions, including weather events and transportation delays that can significantly impact food availability and quality. This technology allows for proactive decision-making to address potential challenges.

- **AI-Based Quality Control:**


Integrating artificial intelligence into the system for quality control, utilizing image recognition and data analysis to inspect and identify quality issues in

agricultural products. AI-based quality control enhances the efficiency and accuracy of food quality assessment.

These ideation and brainstorming concepts contribute to the vision of the "Blockchain – Based Agriculture Data Management System," aiming to deliver a comprehensive, secure, and user-friendly solution for agriculture data management and transparency.




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

Template




Agriculture Data Management

Conducting an effective brainstorming session is not uncommon; however, a successful brainstorm in our agricultural project is unique. It sets the stage for innovative thinking, harnessing the power of blockchain technology and collaborative input to drive the project forward with groundbreaking ideas that will revolutionize data management in the agriculture sector


 15 minutes to prepare
 30-60 minutes to collaborate
 3-8 people recommended


SUTHARSAN T (TEAM LEADER)
JENIL JEBA EPHRAIM J
MANOJ THANGA BALAJI S
NISHANTH K
SIVA ARUN KUMAR S





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

 Team Gathering


 Set Goals





Define your Problem Statement

Design a smart contract using the Ethereum blockchain where you can add the relevant documents on agriculture data into the blockchain You should be able to add the agriculture product details into the blockchain, should be able to query the details from the blockchain, and then change the details whenever it is required.

 10 minutes

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm solo

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

⌚ 10 minutes

Person 1

Transition to blockchain-based data management.

Small-scale rural farmers with limited access to technology.

Person 5

Managing small agricultural operations with resource constraints.

Primary producers responsible for crop growth and supply chain initiation.

Person 2

Efficiently track and share product data.

Decide to adopt blockchain technology.

Person 3

Data security breaches in the blockchain system.

Data security breaches in the blockchain system.

Person 4

Efficient and secure data management.

Improved profitability and agricultural efficiency.

3

Brainstorm as a group

Have everyone move their ideas into the "group sharing space" within the template and have the team silently read through them. As a team, sort and group them by thematic topics or similarities. Discuss and answer any questions that arise. Encourage "Yes, and..." and build on the ideas of other people along the way.

⌚ 15 minutes

TIP
You can use the Voting session tool now to focus on the strongest ideas.

Step-3: Idea Prioritization

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 can break it up into smaller sub-groups.

⌚ 20 minutes

TIP
Add customer stories to sticky notes to add context to them. Consider organizing and categorizing important ideas as they arise within your report.

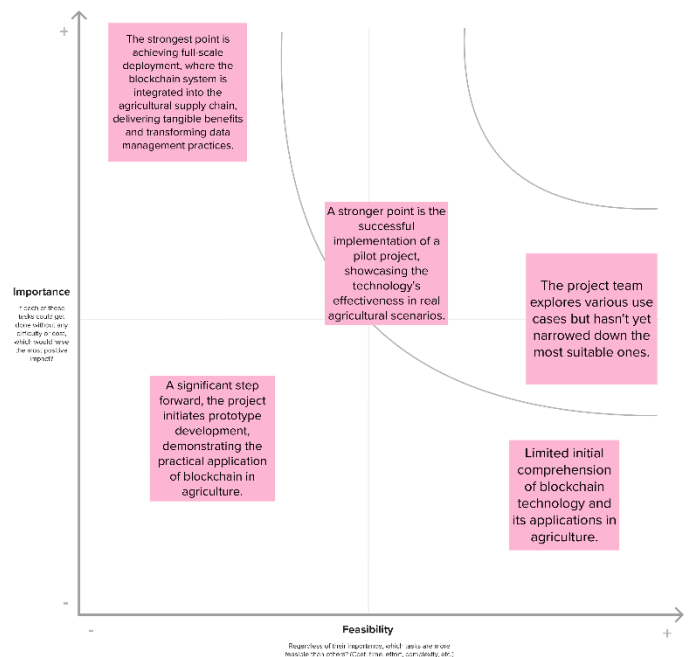
4

Prioritize

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

⌚ 20 minutes

TIP
Participants can use their devices to collect and share sticky notes. Make sure to go on the grid. The facilitator can confirm if a sticky note is being shared by using the "share" button on the keyboard.



CHAPTER 4
REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

FR NO	FUNCTIONAL REQUIREMENT	SUB REQUIREMENT
FR – 1	User Registration and Authentication	Users should be able to create accounts with unique usernames and passwords. Users should have the option for secure authentication methods (e.g., two-factor authentication).
FR – 2	Agriculture Data Recording	Users can input agriculture product details, including crop information, livestock data, and supply chain information. - Data should be stored securely on the blockchain.
FR – 3	Querying Agriculture Data	Users should be able to query agriculture data, filter information by various parameters, and retrieve specific details from the blockchain.

FR – 4	Data Update and Modification	Users should have the capability to update and modify agriculture data when necessary, ensuring that the blockchain remains accurate and up-to-date.
FR – 5	Data Privacy and Security	Implement robust data security measures to protect sensitive agriculture information from unauthorized access and tampering.
FR – 6	User Interface	Develop a user-friendly interface that allows users to interact with the smart contract and blockchain easily.
FR - 7	Role Based Access Control	Implement role-based access control to manage user permissions for data modification.
R - 8	Audit Trial	Maintain an audit trail of data changes to enhance transparency and traceability within the blockchain.

These functional requirements are designed to ensure that the "Blockchain – Based Agriculture Data Management System" is capable of recording, querying, and securely managing agriculture data on the Ethereum blockchain.

4.2 NON-FUNCTIONAL REQUIREMENTS:

NFR NO	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR – 1	Scalability	The system should be designed to easily scale to accommodate an increasing user base and growing data.
NFR – 2	Reliability	The system should be available and reliable, with minimal downtime or service interruptions.
NFR – 3	Security	Data should be stored securely, with encryption to protect sensitive user information.
NFR – 4	Data Backup and Recovery	Regular data backups should be performed to prevent data loss in the event of system failures.
NFR – 5	Usability	The system should be user-friendly, with an intuitive and easy-to-navigate interface.
NFR – 6	Accessibility	The system should comply with accessibility standards (e.g., WCAG) to make it usable by individuals with disabilities.

NFR – 7	Response Time	Define acceptable response times for various system operations (e.g., loading a page, submitting data).
---------	---------------	---

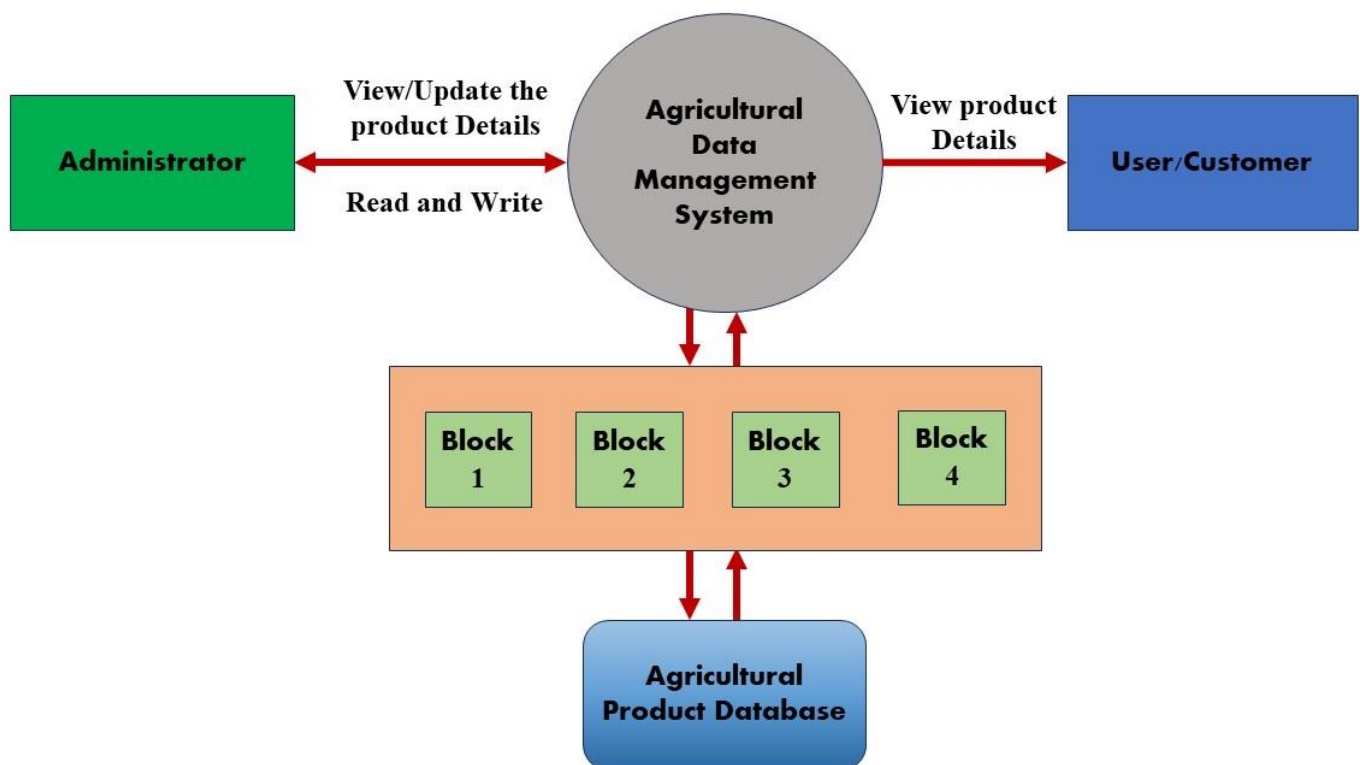
These non-functional requirements outline the performance, reliability, security, and usability standards that the "Blockchain – Based Agriculture Data Management System" should adhere to in order to provide a robust and user-centric solution.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS & USER STORIES

DATA FLOW DIAGRAM:



BLOCKCHAIN – BASED AGRICULTURE DATA MANAGEMENT SYSTEM

USER STORIES:

User Story 1: As a Farmer, I want to upload and update information about my agricultural products in the blockchain database, so I can maintain accurate and up-to-date records.

Requirements:

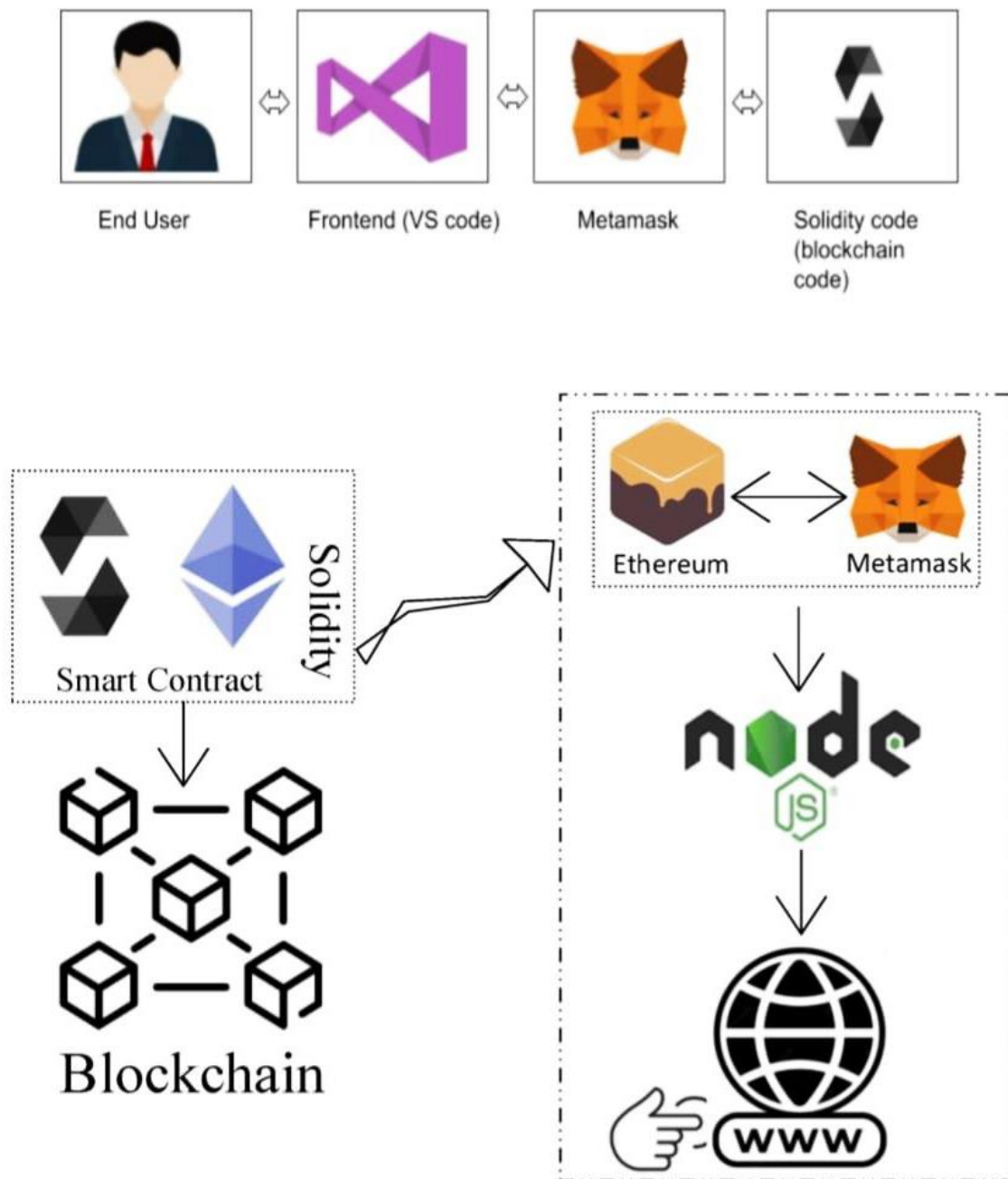
- The system should allow me to upload details of my agricultural products, including crop information, livestock data, and supply chain information.
- I should be able to easily update this information when there are changes, such as crop growth stages, new livestock data, or supply chain updates.
- The system should maintain a secure and immutable record of the information I upload, ensuring data accuracy and traceability.

User Story 2: As a Stakeholder in the Agricultural Industry, I want to view specific details of an agricultural product stored in the blockchain database, so I can gather information about a particular product.

Requirements:

- The system should provide a search or query feature that allows me to specify the product I'm interested in.
- When I search for a specific agricultural product, the system should display detailed information, including crop or livestock details, supply chain history, and any relevant data associated with the selected product.
- The information should be presented in a clear and organized manner, allowing me to quickly access the data I need for decision-making or verification.

5.2 SOLUTION ARCHITECTURE

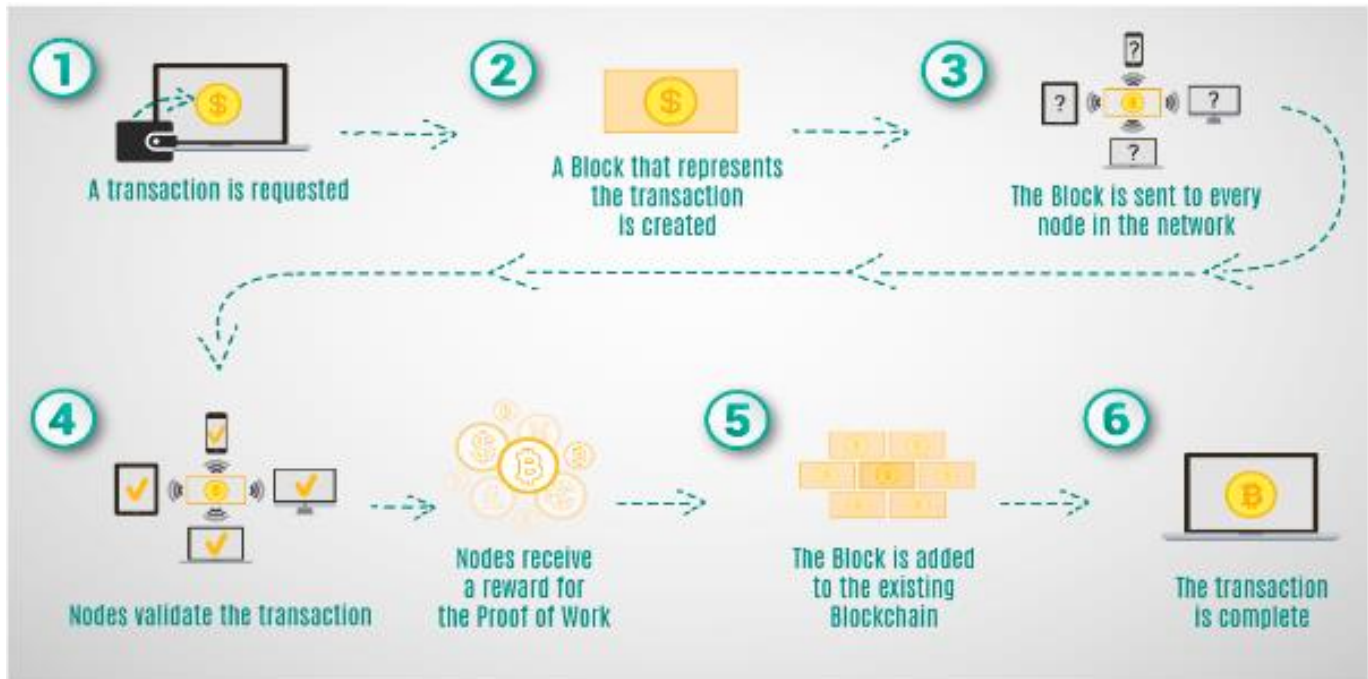


Interaction between the Web and the Contract

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECTURE



6.2 SPRINT PLANNING & ESTIMATION

Sprint Planning:

Sprint planning is an integral part of our agile development process, ensuring that the project team selects and commits to completing specific work items from the product backlog during the upcoming sprint.

Reviewing Product Backlog: Our project team, composed of the Product Owner and the development team, conducts regular product backlog reviews. During these reviews, we evaluate user stories and technical tasks, taking into account the evolving needs and priorities of the project. This ongoing evaluation ensures that our work aligns with the project's overall goals.

Setting Sprint Goals: Based on the product backlog, our team establishes clear sprint goals. These goals serve as a guiding compass for the team throughout the sprint, ensuring alignment with the broader project

objectives. Sprint goals help us maintain a focus on what needs to be achieved within the sprint.

Breaking Down User Stories: User stories and tasks are further decomposed into smaller, actionable sub-tasks. This detailed breakdown provides a comprehensive plan for the sprint, making it easier to understand the specific steps required to accomplish each task. Breaking down user stories helps ensure that we can tackle each part efficiently.

Estimating Work: Our development team employs agile estimation techniques, such as story points, to estimate the effort required for each task. These estimates help the team understand the scope and complexity of the work to be completed during the sprint. Accurate estimation is crucial for effective sprint planning and resource allocation.

Sprint Backlog: The selected user stories and tasks, along with their corresponding estimates, collectively form the sprint backlog. This backlog serves as the foundation for what the team will work on during the sprint. It includes a clear list of tasks to be completed and their associated estimates, ensuring that the team is aligned on the work to be accomplished.

This sprint planning and estimation process enables our team to effectively organize and execute the development of the "Blockchain-Enhanced Agriculture Data Management System." It ensures that we can allocate our resources efficiently and work towards achieving our project goals within each sprint.

Estimation Techniques:

In our project, we employ several estimation techniques to ensure that we can accurately gauge the complexity and effort required for tasks within each sprint. These techniques are vital for effective planning and resource allocation:

Story Points: Story points serve as a relative measure of the complexity and effort needed to complete a task. Tasks are assigned story point values based on their complexity compared to reference tasks. In the context of our project, story points help us assess the complexity of user stories and technical tasks related to agriculture data management. By using this technique, we can prioritize and allocate resources efficiently based on the estimated effort required for each task. Story points provide a common language for our development team to understand the relative complexities of different tasks.

Adjustment for Uncertainty: Recognizing the inherent uncertainty in complex projects, we incorporate an adjustment for uncertainty in our estimation process. Complex projects, like the development of the "Blockchain-Enhanced Agriculture Data Management System," may encounter unforeseen challenges, changing requirements, or unexpected issues. To accommodate such uncertainties, we establish buffers and contingencies within our sprint planning. These buffers allow us to address any unexpected deviations from the original plan while maintaining our commitment to sprint goals and timelines. This adjustment ensures that we can adapt to changing circumstances and still deliver a high-quality product within the defined project constraints.

6.3 SPRINT DELIVERY SCHEDULE

Sprint Delivery Schedule for Food Tracking System Project

Sprint Delivery Schedule and its Objectives		
Sprint 1	Project Initiation	Set project scope and team roles.
Sprint 2	Data Modeling	Define data models and prioritize user stories.
Sprint 3	Blockchain Integration	Begin integrating blockchain technology.
Sprint 4	User Interfaces and Mobile Apps	Develop user interfaces and mobile apps.
Sprint 5	Quality Control and Testing	Implement quality control and initiate testing.
Sprint 6	Compliance and Security	Ensure regulatory compliance and enhance security.
Sprint 7	Deployment Preparation	Prepare for system deployment.
Sprint 8	Pilot Testing and Optimization	Conduct pilot tests and optimize the system.
Sprint 9	Full System Deployment	Roll out the fully operational system to a wider audience.

This sprint delivery schedule outlines the specific objectives and focus areas for each sprint of the "Blockchain-Enhanced Agriculture Data Management System" project. It ensures that the project progresses systematically, with clear goals for each sprint.

CHAPTER 7

CODING & SOLUTIONING

Smart Contract (Solidity)

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract AgricultureRegistry {

struct foodProduct {

string name;

string description;

uint256 quantity;

address owner;

}

mapping(uint256 => foodProduct) public products;

uint256 public productCount;

event ProductAdded(uint256 productId, string name, string description,
uint256 quantity, address owner);

event ProductUpdated(uint256 productId, string name, string
description, uint256 quantity);

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 key features related to food tracking in this contract are:

7.1 FEATURE 1

➤ Agriculture Product Registration and Update:

The smart contract allows for the registration and updating of agricultural products. Users can add information about an agricultural product, including its name, description, and quantity. This feature is essential for tracking and recording data about various agricultural products within the blockchain. Users, such as farmers or suppliers, can update the product information as needed, ensuring that the data remains accurate and up to date.

7.2 FEATURE 2

➤ Ownership Control:

The contract includes an ownership control mechanism. Only the owner of a particular product (as specified by the **‘owner’** field) can perform actions such as updating the product's information. This feature enhances data security and ensures that only authorized users can modify the information associated with specific products.

7.3 DATABASE SCHEMA

In the context of the "Blockchain-Enhanced Agriculture Data Management System," the database schema is primarily defined by the structure of the smart contract deployed on the Ethereum blockchain. The Ethereum blockchain, being a decentralized and distributed ledger, serves as the underlying data storage mechanism for the project, eliminating the need for a traditional relational database schema.

The core data structure within the smart contract is the **‘foodProduct’** struct, which is used to organize and store information about agricultural products.

Here is the structure of the '**foodProduct**' and how data is stored within the contract:

```
struct foodProduct {  
  
    string name;  
  
    string description;  
  
    uint256 quantity;  
  
    address owner;  
  
}
```

- **name:** Represents the name or title of the agricultural product.
- **description:** Contains a description or additional details about the product.
- **quantity:** Indicates the quantity or amount of the product.
- **owner:** Records the Ethereum address of the user who owns the product.

The data for each agricultural product is stored within the **products** mapping using a unique product identifier as the key. The Ethereum blockchain provides the underlying storage for this data, and it is inherently decentralized and immutable. There is no need for traditional relational database tables, as the blockchain itself acts as the distributed ledger for maintaining the integrity and security of the agricultural data.

This decentralized and blockchain-based database schema ensures that agricultural data is transparent, tamper-proof, and securely managed within the system. Users can access and update data with confidence, and the blockchain's architecture guarantees data immutability and traceability.

CHAPTER 8

PERFORMANCE TESTING

8.1 PERFORMANCE METRICS

In the evaluation of the "Blockchain-Enhanced Agriculture Data Management System," several performance metrics are crucial for assessing the system's effectiveness and efficiency. These metrics are instrumental in ensuring the system's ability to meet the requirements of a modern and secure agricultural data management solution:

1. Transaction Throughput:

- **Metric:** Measure the number of transactions the blockchain can process per second.
- **Importance:** Higher throughput is essential for handling a large volume of data in real-time, ensuring the system can accommodate the demands of the agricultural supply chain.

2. Data Accuracy:

- **Metric:** Assess the accuracy of the data recorded on the blockchain.
- **Importance:** Ensure that the system minimizes errors and discrepancies in the supply chain data, enhancing trust and reliability in the recorded information.

3. Data Integrity:

- **Metric:** Measure the immutability of data stored on the blockchain.

- **Importance:** Verify that once data is recorded, it cannot be altered or deleted, ensuring data integrity and preventing unauthorized modifications.

4. Security:

- **Metric:** Monitor the system's ability to protect sensitive data and prevent unauthorized access or tampering.
- **Importance:** Maintaining robust security is crucial for safeguarding confidential agricultural data and ensuring the privacy of stakeholders.

5. Response Time:

- **Metric:** Evaluate the time it takes for the system to respond to inquiries or requests, such as tracking a product's origin.
- **Importance:** A prompt response time is critical for users to access timely information, enhancing the system's usability and efficiency.

6. Supply Chain Efficiency:

- **Metric:** Assess the system's impact on the overall efficiency of the food supply chain.
- **Importance:** Evaluate how the system contributes to reducing waste, optimizing inventory management, and minimizing delays in the agricultural supply chain.

7. Feedback and Improvement:

- **Metric:** Collect feedback from users and stakeholders to continuously improve the system, including the user interface and features.

- **Importance:** Ongoing feedback and improvements are essential for ensuring the system aligns with user needs and evolving industry requirements.

8. Cost Efficiency:

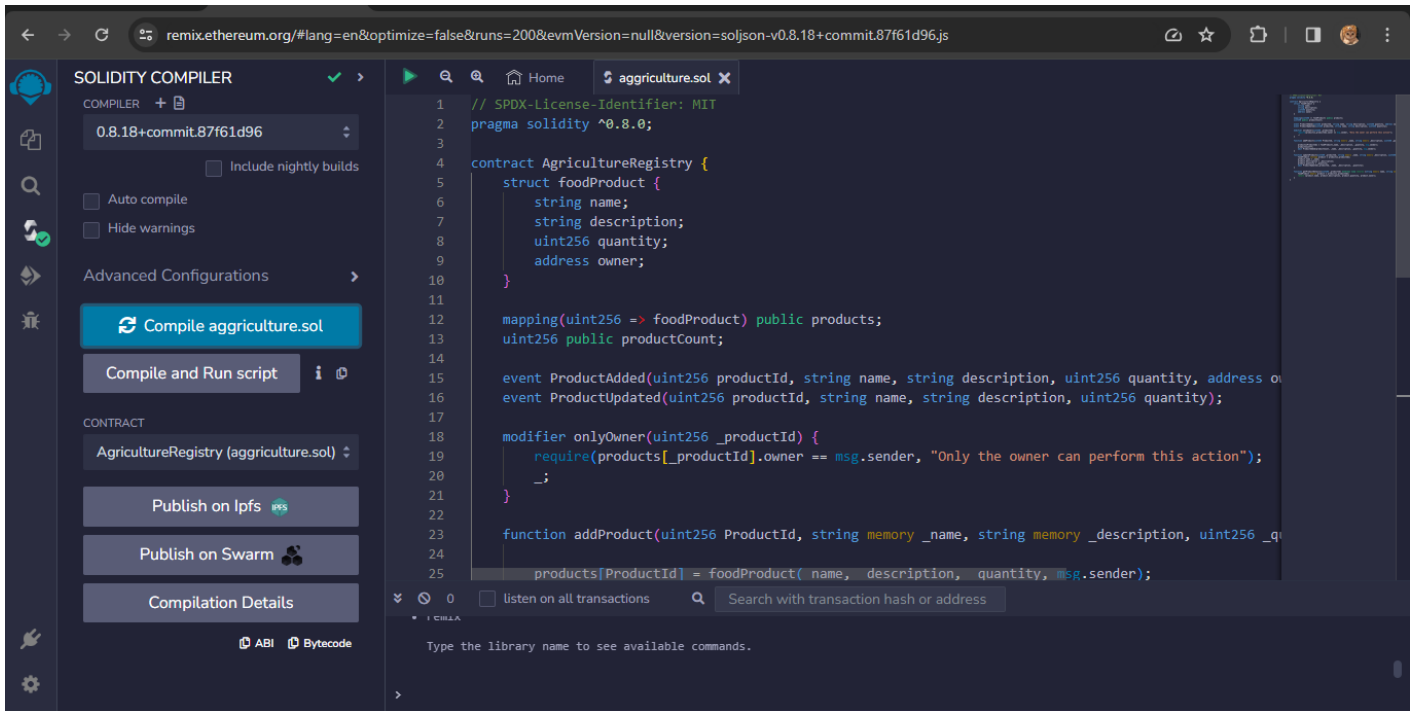
- **Metric:** Assess the system's cost-effectiveness, including the cost of blockchain technology, IoT sensors, and maintenance compared to the benefits it provides.
- **Importance:** Evaluating cost efficiency helps ensure that the system provides value for its investment and operational costs.

These performance metrics guide the evaluation and optimization of the "Blockchain-Enhanced Agriculture Data Management System," ensuring that it meets the requirements of a modern and efficient agricultural data management solution.

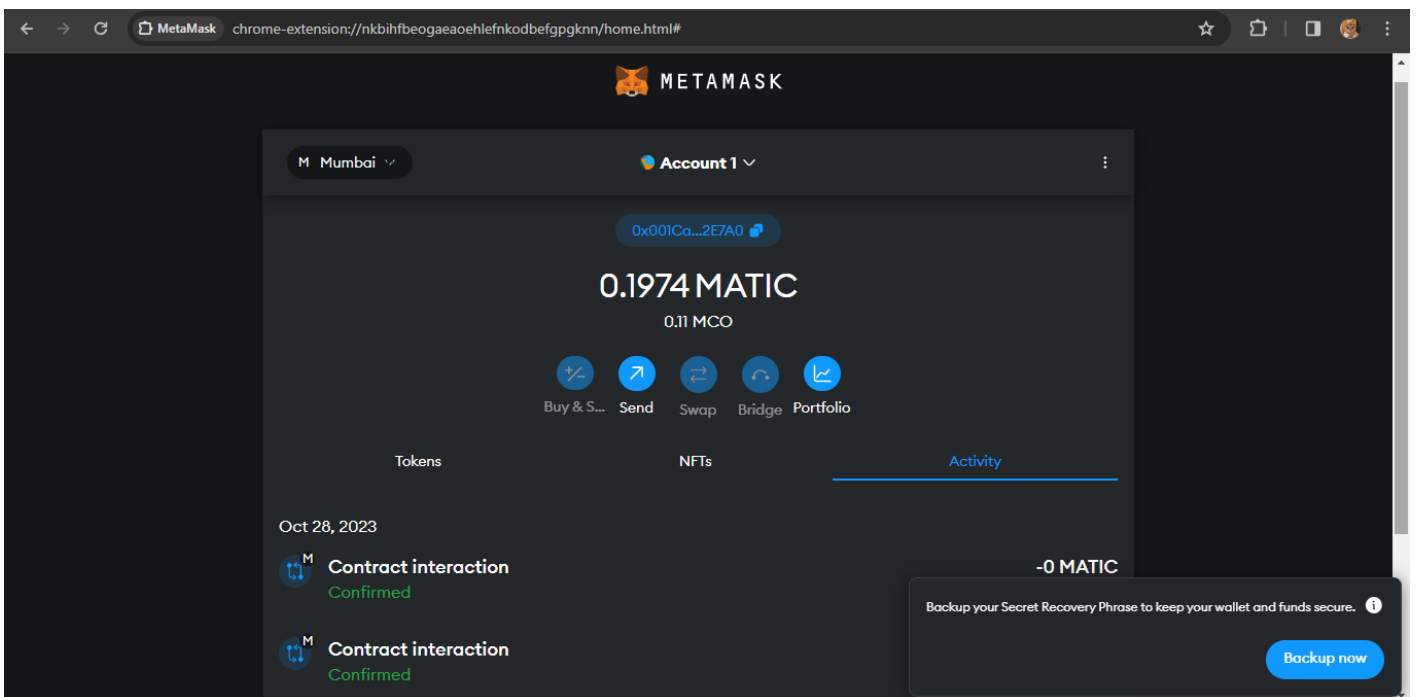
CHAPTER 9

RESULTS

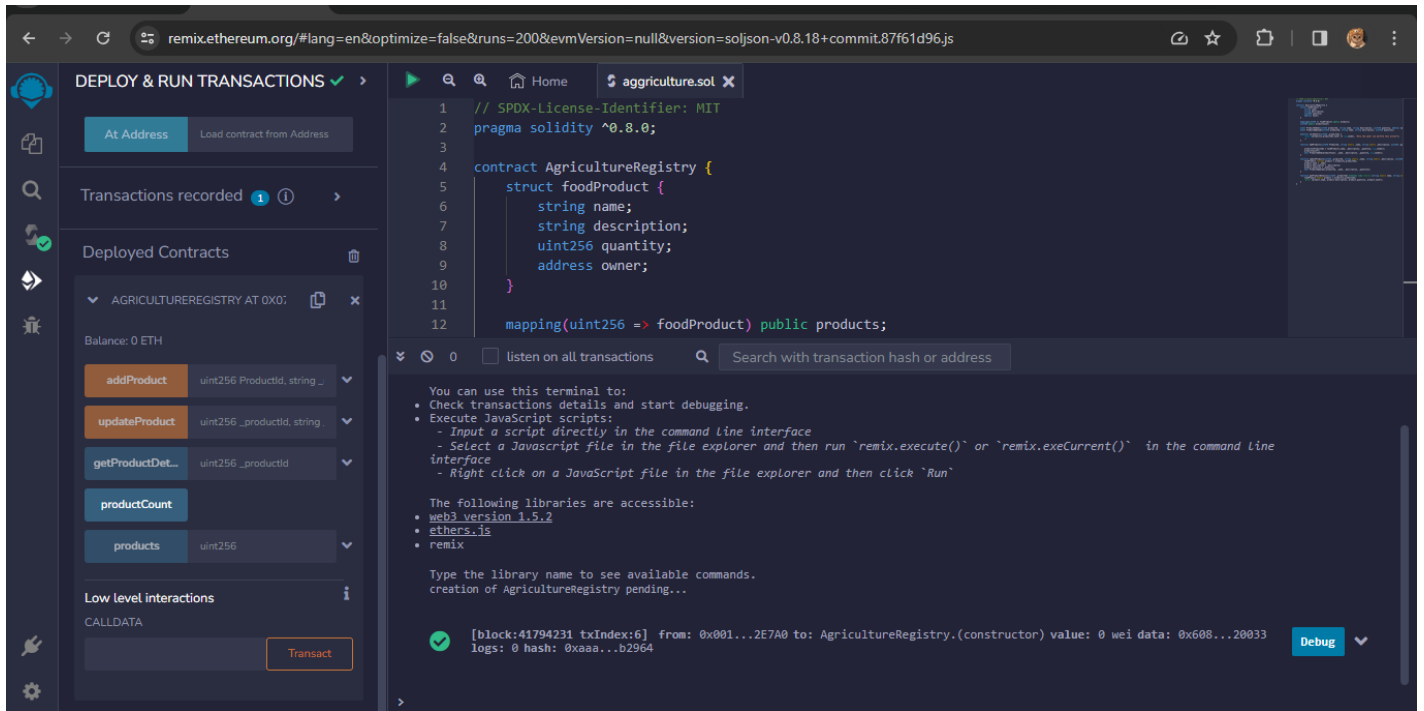
9.1 OUTPUT SCREENSHOTS



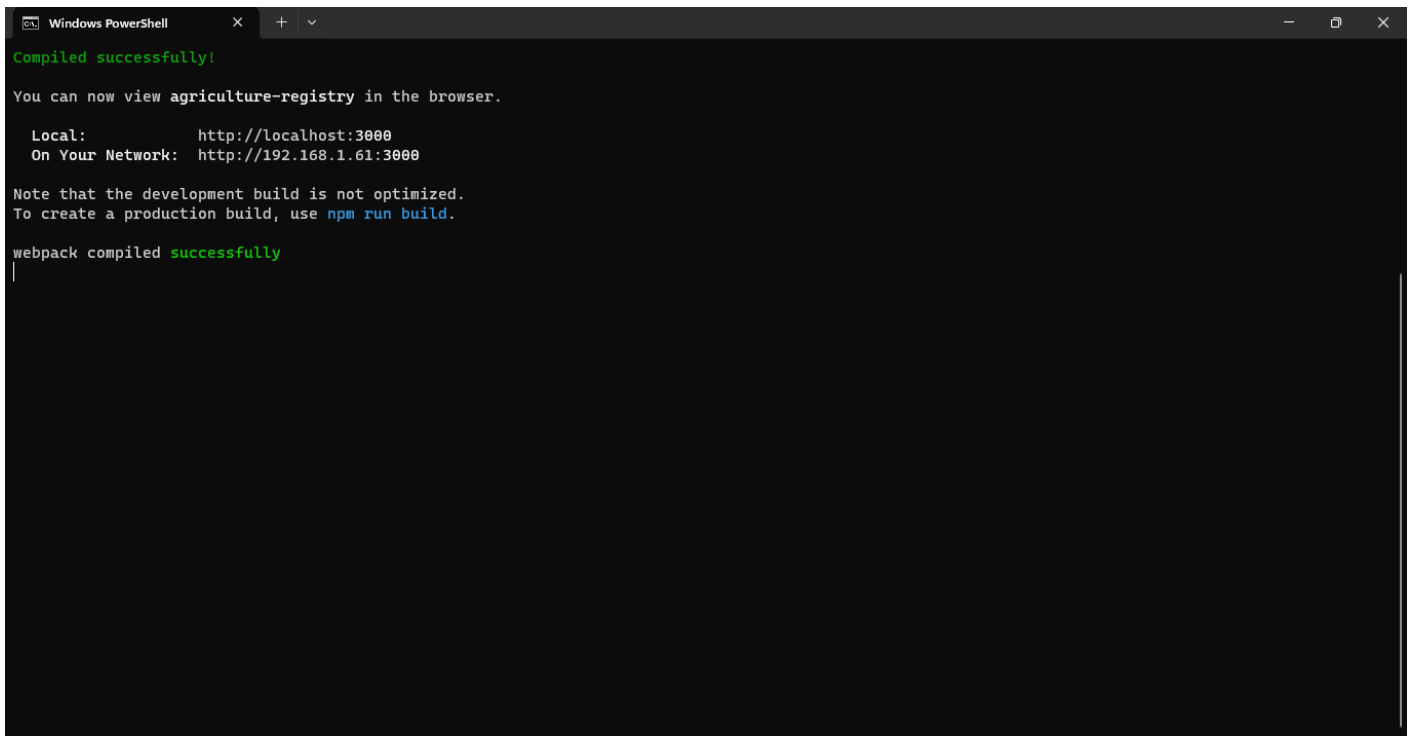
REMIX IDE – CREATING A SMART CONTRACT



METAMASK ACCOUNT



REMIX IDE – DEPLOYED CONTRACT



TERMINAL OUTPUT SCREEN

localhost:3000

Agriculture Registry

0x001c....42e7a0

Product Id	Product Id
Product Name	Product Name
product description	product description
product quantity	product quantity
<button>Add Product</button>	<button>Update Product</button>

Enter Id

FINAL FRONTEND OUTPUT SCREEN

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

1. **Data Transparency:** Blockchain ensures transparent and immutable records, enhancing trust and transparency in the agricultural supply chain.
2. **Traceability:** Users can trace the origin and journey of agricultural products, promoting food safety and quality assurance.
3. **Data Security:** The decentralized nature of blockchain enhances data security, protecting sensitive information from unauthorized access or tampering.
4. **Efficiency and Reduced Waste:** Improved supply chain efficiency reduces waste and optimizes inventory management, resulting in cost savings.
5. **User Engagement:** The system's user-friendly interface and features promote user engagement and participation in data management.

DISADVANTAGES:

1. **Cost of Implementation:** The initial setup, including blockchain technology and potential IoT sensors, can be costly, impacting the project's budget.
2. **Technological Barriers:** Some stakeholders may face challenges in adopting and adapting to blockchain technology and IoT sensors.
3. **Regulatory Compliance:** Ensuring compliance with relevant regulations and standards can be complex and time-consuming.

4. **Scalability:** Scaling the system to accommodate a growing volume of data and users may pose challenges that require ongoing maintenance and optimization.

5. **Data Accuracy:** Relying on user-input data can lead to inaccuracies or inconsistencies in the recorded information, affecting data reliability.

CHAPTER 11

CONCLUSION

CONCLUSION:

In conclusion, the "Blockchain-Enhanced Agriculture Data Management System" represents a significant advancement in the agricultural industry, offering numerous benefits for stakeholders across the supply chain. The project's primary objective was to leverage blockchain technology to enhance data transparency, traceability, and security, ultimately contributing to the efficiency and reliability of the agricultural supply chain.

Throughout the project, we have addressed the challenges associated with traditional data management systems, such as data accuracy, transparency, and security. By adopting blockchain as the foundational technology, we have ensured that agricultural data is stored in a decentralized and immutable manner, eliminating risks of data manipulation and enhancing trust among stakeholders.

CHAPTER 12

FUTURE SCOPE

FUTURE SCOPE:

As the "Blockchain-Enhanced Agriculture Data Management System" continues to evolve and adapt to the dynamic landscape of the agricultural industry, several exciting future prospects and advancements are on the horizon. These possibilities represent opportunities to further enhance the system's capabilities and offer even greater benefits to stakeholders:

1. Integration with AI and Machine Learning:

Future systems will increasingly leverage AI and machine learning to provide more personalized recommendations and insights based on individual health goals, preferences, and dietary restrictions. These intelligent algorithms can analyze vast datasets to offer tailored nutritional advice and optimize food management.

2. Wearable Technology Integration:

Integration with wearable technology, such as smartwatches and fitness trackers, will enable real-time monitoring of food intake. Users can effortlessly track their dietary habits through wearable devices, ensuring accurate and continuous data collection for improved health and nutrition management.

3. Augmented Reality (AR) Applications:

AR applications and visual recognition technology may revolutionize food tracking by allowing users to capture and identify foods using their smartphones. This technology streamlines the tracking process, making it

more efficient and user-friendly. Users can simply point their devices at a meal to log its nutritional information.

4. Enhanced Blockchain Traceability:

Blockchain technology can extend its utility beyond data management and be used to track the entire journey of food from farm to table. This extended use ensures transparency and traceability in the food supply chain, enhancing food safety and quality assurance for consumers and stakeholders.

5. Integration with Healthcare Systems:

Greater integration of food tracking systems with electronic health records and telehealth platforms will enable healthcare providers to monitor and support patients more effectively. The exchange of dietary data and health information can contribute to personalized healthcare plans and better patient outcomes.

The future scope of the "Blockchain-Enhanced Agriculture Data Management System" is promising, with potential advancements that can further empower stakeholders in the agricultural supply chain. These developments will not only enhance data management but also promote healthier living and improved food safety and quality in the industry.

CHAPTER 13

APPENDIX

SOURCE CODE

agricultureOnBlockchain.sol

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

```
pragma solidity ^0.8.0;
```

```
contract AgricultureRegistry {
```

```
    struct foodProduct {
```

```
        string name;
```

```
        string description;
```

```
        uint256 quantity;
```

```
        address owner;
```

```
    }
```

```
    mapping(uint256 => foodProduct) public products;
```

```
    uint256 public productCount;
```

```
    event ProductAdded(uint256 productId, string name, string description,  
uint256 quantity, address owner);
```

```
    event ProductUpdated(uint256 productId, string name, string description,  
uint256 quantity);
```



```
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);
```

```
}
```

```
}
```

Connector.js

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

```
const abi = [
```

```
{
```

```
  "anonymous": false,
```

```
  "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"
  },
  {
    "indexed": false,
    "internalType": "address",
    "name": "owner",
    "type": "address"
  }
],
"name": "ProductAdded",
"type": "event"
},
```

```
{  
  "anonymous": false,  
  "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",  
      "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": [
    {
      "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",  
  "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",  
  "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('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 = "0xE3E79289C9dcD24DD6F591C15bcD74FBA532d3A3"  
  
export const contract = new ethers.Contract(address, abi, signer)
```

home.js

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

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

```
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)
```

```
}
```

```
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())

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

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

```
}
```

```
const handleWallet = async () => {
```

```
  if (!window.ethereum) {
```

```
    return alert('please install metamask');
```

```
  }
```

```
  const addr = await window.ethereum.request({
```

```
    method: 'eth_requestAccounts',
```

```
  });
```

```
  setWallet(addr[0])
```

```
}
```

```
return (
```

```
<div>
```

```
  <h1 style={{ marginTop: "30px", marginBottom: "80px" }}>Agriculture  
  Registry</h1>
```

```
  {!Wallet ?
```

```
    <Button onClick={handleWallet} style={{ marginTop: "30px",  
    marginBottom: "50px" }}>Connect Wallet </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>
```

```
<input style={{ 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 />
```

```
<input style={{ 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>
```

```
<Col style={{ 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="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 >
```

```
    <div style={{ margin: "auto" , marginTop:"100px"}}>
```

```
      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleGetIds} type="number" placeholder="Enter Id"
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>

)
}

```

```
export default Home;
```

App.js

```

import './App.css';
import Home from './Page/Home'

function App() {
    return (
        <div className="App">
            <header className="App-header">
                <Home />
            </header>
        </div>
    );
}

```

```
}
```

```
export default App;
```

index.js

```
import React from 'react';
```

```
import ReactDOM from 'react-dom/client';
```

```
import './index.css';
```

```
import App from './App';
```

```
import reportWebVitals from './reportWebVitals';
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(  
  <React.StrictMode>
```

```
    <App />
```

```
  </React.StrictMode>
```

```
);
```

```
// If you want to start measuring performance in your app, pass a function
```

```
// to log results (for example: reportWebVitals(console.log))
```

```
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

```
reportWebVitals();
```

reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {  
  if (onPerfEntry && onPerfEntry instanceof Function) {  
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB })  
=> {  
      getCLS(onPerfEntry);  
      getFID(onPerfEntry);  
      getFCP(onPerfEntry);  
      getLCP(onPerfEntry);  
      getTTFB(onPerfEntry);  
    });  
  }  
};  
  
export default reportWebVitals;
```

setupTests.js

```
import '@testing-library/jest-dom';
```

GITHUB & PROJECT DEMO LINK

- GitHub Link: <https://github.com/Sutharsan56/NM2023TMID06191.git>
- Demo Link: <https://drive.google.com/file/d/1X3mljE10SvoUXI03ppgwl4LY3H1EtrD/view?usp=sharing>