PROJECT REPORT

1. Introduction:

Project Name: Food Tracking System

1.1 Project Overview:

The Food Tracking System is a digital application designed to help individuals monitor their dietary intake, make informed nutritional choices, and maintain a healthy lifestyle. This system aims to simplify the process of recording, analysing, and visualizing food consumption data, promoting better eating habits and overall well-being. Develop a user-friendly mobile or web-based application for tracking food consumption .User Registration and Profile Management

1.2Purpose

1. Block chain provides an immutable and transparent ledger that records every step of the food supply chain. This enables precise tracking of the journey of food products from farm to fork. It's especially valuable in the event of food safety recalls, as it can quickly identify the source of contamination.
2. The purpose of a food tracking system based on block chain technology is to create a secure, transparent, and efficient food supply chain that benefits all stakeholders, from producers to consumers. It addresses critical issues related to food safety, traceability, fraud prevention, and data integrity while promoting trust and accountability throughout the food industry.

2. Literature Survey

2.1Existing Problem

Data Standardization: Data from various sources within the food supply chain may not be standardized or compatible with each other. Achieving uniformity in data formats and structures is essential for seamless integration with block chain systems.

Integration with Legacy Systems: Many companies in the food industry still rely on legacy systems and may face difficulties when integrating block chain technology into their existing processes. Ensuring interoperability with these systems can be a challenge.

Cost and Scalability: Developing and maintaining a block chain network can be costly, especially for smaller producers and suppliers. Additionally, as more participants join the network, the block chain must scale to handle a higher volume of transactions, which can be technically challenging and resource-intensive.

User Adoption: Encouraging all participants in the supply chain to adopt and consistently use the block chain system can be challenging. Resistance to change and a lack of technical expertise can hinder adoption.

Privacy and Security: While block chain is generally secure, there are still vulnerabilities in the systems and applications built on top of it. Protecting sensitive data and ensuring the privacy of participants is a concern, especially in a public block chain network.

Regulatory Compliance: Food tracking systems must adhere to various local and international regulations. Complying with these regulations while maintaining transparency and security can be complex.

Scalability: As more data is added to the block chain, scalability becomes a concern. Block chain networks must be able to handle an ever-increasing volume of transactions without compromising performance or decentralization.

2.2References

Search for research papers and articles in academic journals related to "food tracking block chain," "block chain in food supply chain," or "block chain for food traceability." Use academic databases like Google Scholar to find research papers and publications that explore the use of block chain for food tracking and supply chain transparency

Search for books on the subject of "block chain in food tracking" or "food supply chain block chain" through online bookstores and libraries. Explore websites and blogs dedicated to block chain technology and the food industry. These sources often provide insights and case studies on the use of block chain for food tracking

Look for papers and presentations from conferences and events focused on block chain technology and the food industry

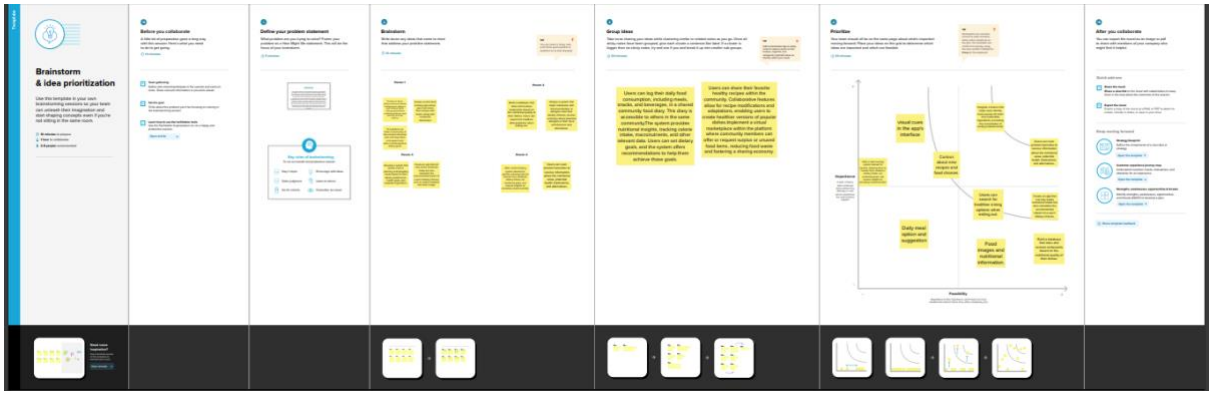
2.3Problem Statement definition

To address these challenges, a new food tracking system is needed that leverages emerging technologies, such as block chain and IoT, to enhance transparency, accuracy, and security while providing real-time traceability and comprehensive information to consumers and businesses.

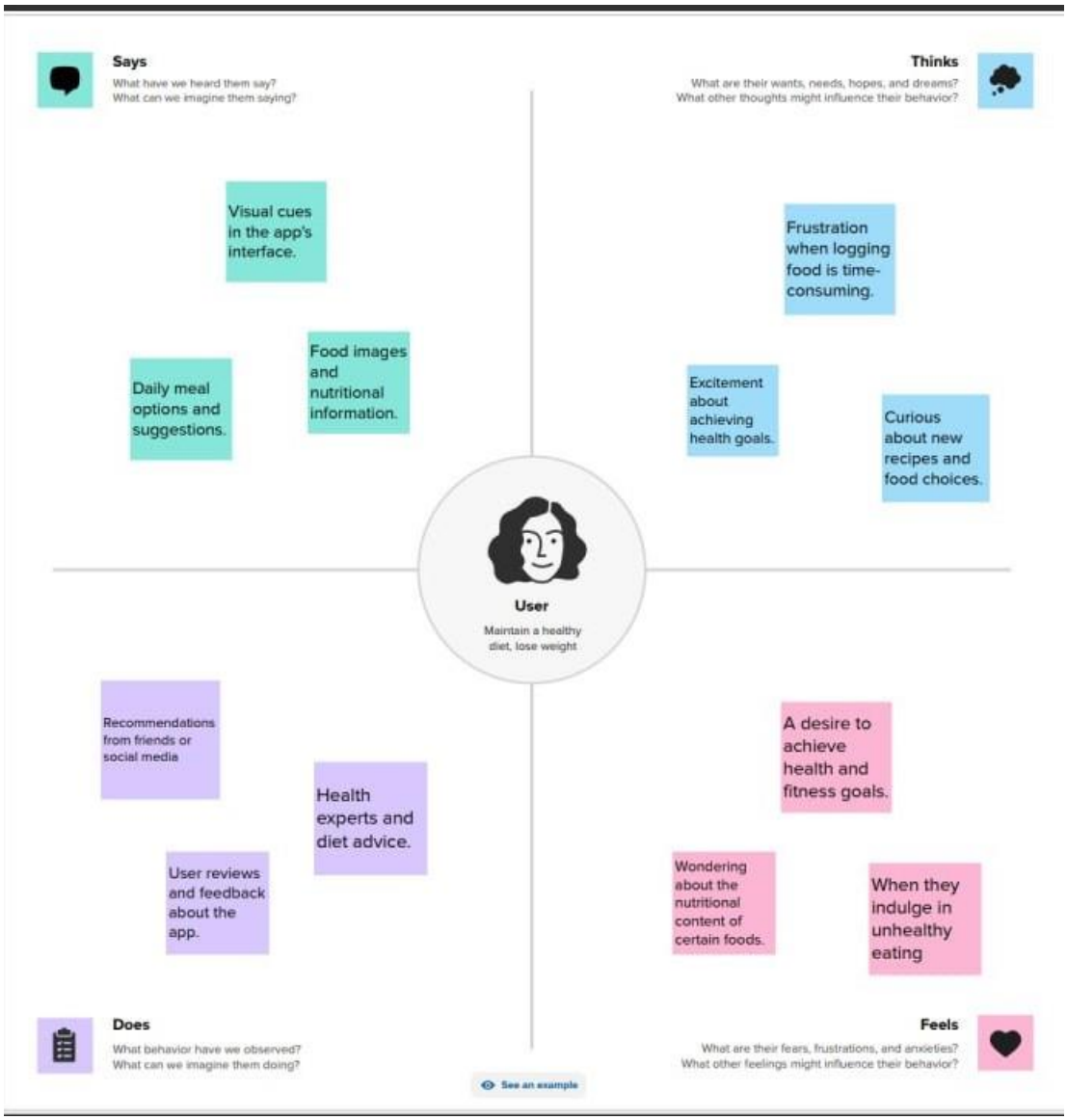
This system aims to revolutionize food tracking by promoting trust, safety, and sustainability throughout the food supply chain.

3.Ideation and Proposed Solution

3.1 Empathy Map Canvas



3.2Ideation and Brainstorming:



4.Requirement Analysis

4.1Functional Requirements

User Registration and Authentication:

Users should be able to create accounts securely.

Two-factor authentication should be available for added security.

Users can recover their accounts if necessary.

Product Data Entry and Verification:

Producers and suppliers should be able to input detailed information about food products.

Data should include product origin, production methods, expiration dates, and relevant certifications.

Verification processes should ensure the accuracy of data entered into the system.

Block chain Ledger:

The system should maintain a distributed ledger of all food product transactions, recording each step in the supply chain.

Transactions should be immutable, preventing tampering with historical data.

Smart Contracts:

Smart contracts should automate processes such as payment, verification, and quality assurance.

They should execute predefined actions when certain conditions are met.

Food Traceability:

Users (consumers and businesses) should be able to trace the entire supply chain of a food product.

The system should display a complete history of a product's journey from production to consumption.

Product Scanning and QR Codes:

Consumers should be able to scan QR codes or use other methods to access product information easily.

Scanning should provide immediate access to the blockchain data for the scanned product.

Privacy and Data Security:

User data and food tracking information should be securely stored and protected.

Users should have control over the visibility of their data and who can access it.

Real-time Updates:

The system should provide real-time updates on the status and location of food products in transit.

Alerts should be generated in case of delays or anomalies in the supply chain.

Food Recalls and Alerts:

The system should enable rapid identification and recalls of potentially unsafe or contaminated food products.

Notifications should be sent to affected parties.

Compliance and Certification:

Producers and suppliers should be able to upload compliance documents and certifications.

The system should automatically verify compliance with food safety and quality standards.

4.2 Non-Functional Requirements:

Performance:

Response Time: The system should provide quick response times, especially when users are searching for food items or inputting data.

Scalability: The system should be able to handle increased load as the number of users or data entries grows.

Availability:

Uptime: The system should be available 24/7 with minimal downtime for maintenance or upgrades.

Fault Tolerance: The system should be able to continue functioning even in the presence of hardware or software failures.

Reliability:

Data Accuracy: The system should maintain accurate and up-to-date information about food items and their nutritional content.

Backup and Recovery: There should be regular backups and a clear process for data recovery in case of data loss or system failure.

Security:

User Authentication: Users should be required to authenticate themselves to access their data, ensuring data privacy.

Data Encryption: Data transmission and storage should be encrypted to protect against unauthorized access.

Access Control: Different users should have appropriate access rights and permissions based on their roles.

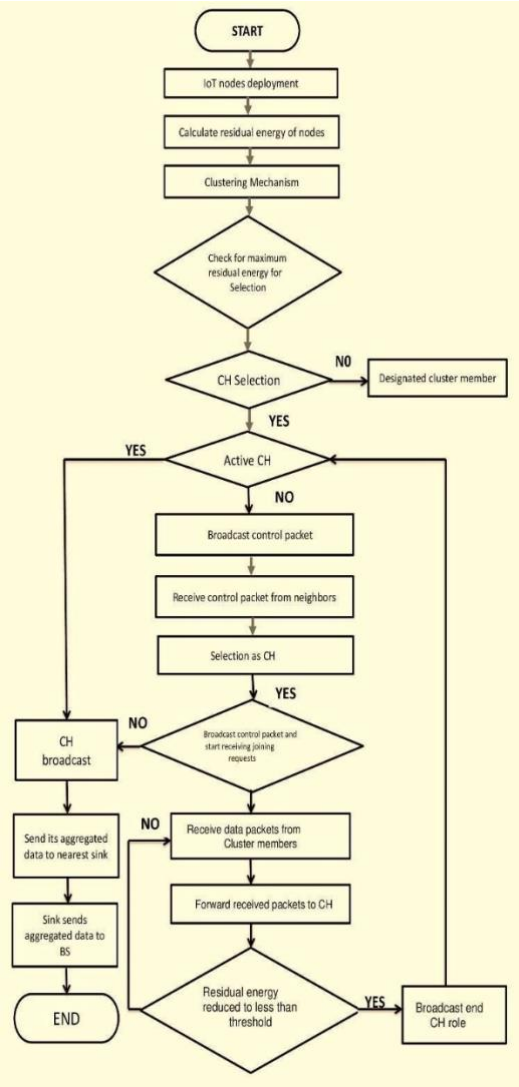
Usability:

User-Friendly Interface: The system should have an intuitive and user-friendly interface for easy navigation and data entry.

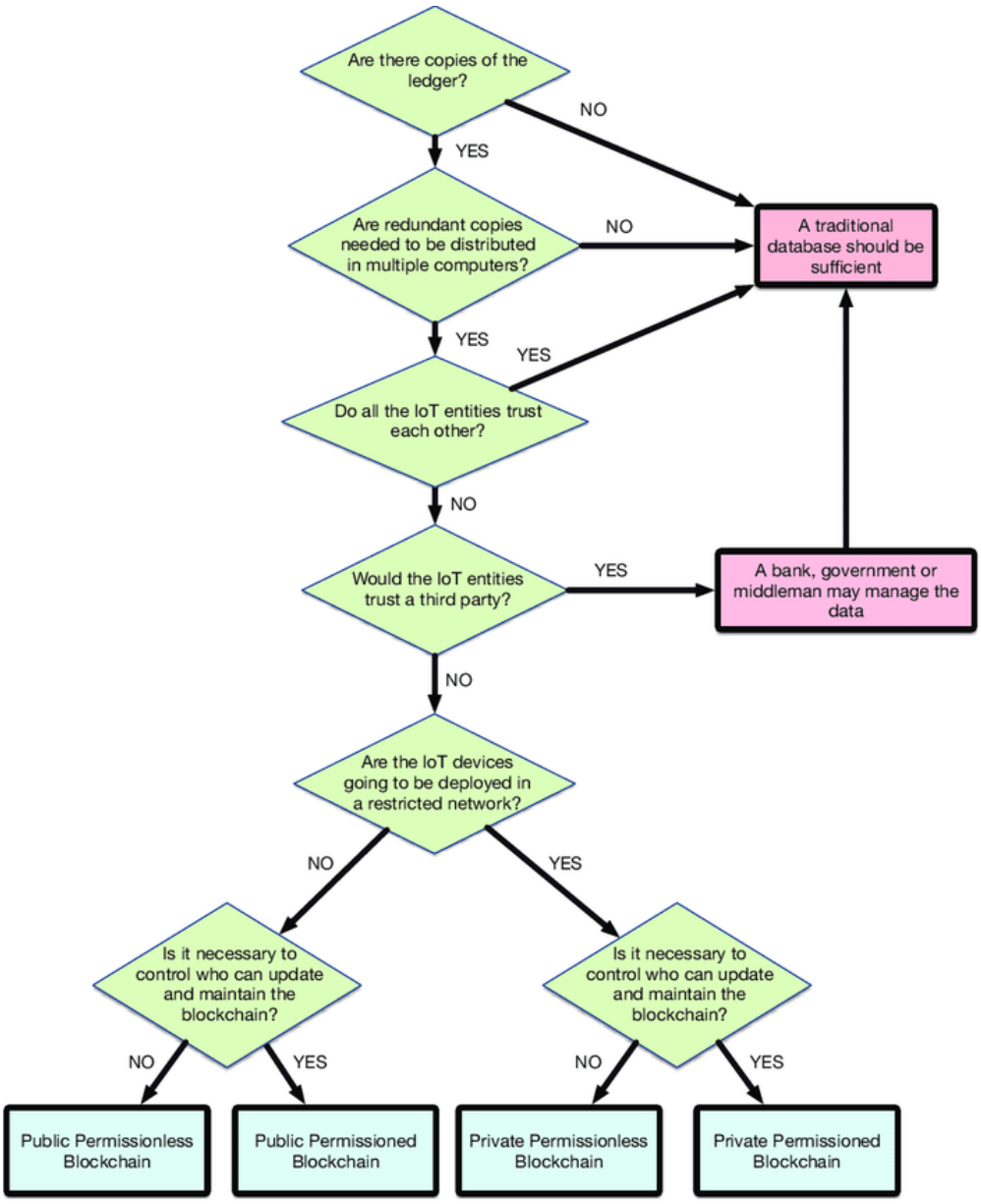
Accessibility: The system should be accessible to people with disabilities in compliance with accessibility standards (e.g., WCAG).

5. Project Design:

5.1 Data flow diagram & User stories:



5.2 Solution Architecture:



6. Project Planning & Scheduling:

6.1 Technical Architecture:

User Interfaces:

Web Application: A user-friendly web interface accessible from desktop and mobile browsers.

Mobile Application: Native mobile apps for Android and iOS platforms to provide a seamless mobile experience.

Application Layer:

Web Server: Handles HTTP requests and serves web pages and RESTful APIs to the clients.

Application Server: Contains the core application logic, including user management, food tracking, data processing, and integration with external services.

Authentication and Authorization: Handles user authentication and access control.

Data Management:

Database Management System (DBMS): Stores and manages the application's data. Common databases include:

User Profile Database: Stores user account information and preferences.

Food Database: Contains data on food items, including nutritional information.

Tracking Database: Records user-specific food tracking data.

External Integrations:

API Integrations: Connects to external services, such as fitness trackers, wearables, and other health data sources.

Third-Party APIs: Utilizes APIs for features like geolocation, barcode scanning, or recipe suggestions.

Caching Layer:

Caching Server: Utilizes in-memory caching to store frequently accessed data to reduce the load on the database and improve performance.

Analytics and Reporting:

Analytics Engine: Gathers data on user activity and provides insights for users, such as progress reports and recommendations.

Reporting Tools: Enables users to generate custom reports on their food tracking and nutritional data.

Security Layer:

Firewall: Protects against unauthorized access and cyberattacks.

Encryption: Secures data in transit and at rest, using technologies like SSL/TLS and data encryption.

Access Control: Implements user roles and permissions for data privacy and security.

Scalability and Load Balancing:

Load Balancer: Distributes incoming traffic across multiple application servers for load distribution.

Auto-Scaling: Automatically adjusts resources based on traffic and load.

DevOps and Infrastructure:

Continuous Integration/Continuous Deployment (CI/CD): Supports automated testing, deployment, and updates.

Containerization: May use containers (e.g., Docker) for portability and scalability.

Cloud Services: Hosted on cloud platforms like AWS, Azure, or Google Cloud for scalability and reliability.

Monitoring and Logging:

Monitoring Tools: Keep an eye on system health, performance, and potential issues.

Logging Framework: Records application and system events for debugging and auditing.

Backup and Recovery:

Regular backups of data to ensure recovery in case of data loss or system failure.

Compliance and Regulation:

Ensure the system complies with relevant regulations, such as GDPR for data privacy or food safety standards.

6.2Spirit Planning & Estimation:

Backlog Refinement: Before sprint planning, make sure you have a well-defined product backlog. This backlog should include user stories, features, and other tasks that need to be addressed in the food tracking system.

Sprint Planning Meeting: Hold a sprint planning meeting with your development team, product owner, and Scrum master (if following Scrum). During this meeting:

Review Prioritized Backlog: Discuss the items in the product backlog, focusing on the highest-priority ones.

Select Sprint Goal: Define a clear sprint goal for the upcoming sprint. For a food tracking system, it could be improving user data entry or adding a new feature like meal planning.

Task Breakdown: Break down the selected user stories into smaller tasks. For example, if one of the user stories is "User can search for food items," tasks might include designing the search UI, implementing the search functionality, and writing tests.

Estimation: Estimate the effort for each task using a unit of estimation like story points or ideal days.

Commitment: The development team commits to completing the selected tasks within the sprint. Make sure the commitment is realistic based on the team's capacity and velocity.

Definition of Done: Define what it means for a task or user story to be considered "done." For example, a task may be considered done when it's developed, tested, documented.

6.3 Spirit Delivery Schedule:

Meal Delivery Schedule:

Define the days and hours during which meal deliveries will be made. This schedule should align with customer preferences and operational capacity.

Delivery Routes:

Plan efficient delivery routes to ensure timely and accurate deliveries. Route optimization can help reduce delivery times and costs.

Order Management:

Implement an order management system that allows customers to place food orders for delivery. This could include mobile apps or online platforms.

Delivery Tracking:

Utilize technology to track the progress of meal deliveries. Real-time tracking and updates can provide customers with visibility into the status of their orders.

Driver Management:

Assign delivery drivers to specific routes and time slots, and ensure they are equipped with the necessary tools and training to make successful deliveries.

Compliance and Food Safety:

Ensure that all deliveries comply with food safety regulations. Implement procedures to maintain the quality and safety of food products during transportation.

Customer Service:

Establish a customer support system to address inquiries, concerns, and issues related to meal deliveries.

Inventory Management:

Maintain accurate inventory records to track the availability of food items and prevent overbooking.

Cost Management:

Manage the costs associated with meal delivery, such as fuel, vehicle maintenance, driver wages, and logistics. Monitor profitability and make adjustments as needed.

Environmental Considerations:

Consider the environmental impact of meal deliveries and explore eco-friendly options for packaging and transportation.

Feedback and Improvement:

Gather feedback from customers and delivery personnel to identify areas for improvement in the delivery process and the food tracking system's overall functionality.

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

7.1 Feature 1

Immutable Ledger: Block chain maintains an immutable ledger of all transactions and data, ensuring that once data is recorded, it cannot be altered or deleted. This feature helps prevent fraud, tampering, and unauthorized changes in the food supply chain.

End-to-End Traceability: Each step of the food supply chain is recorded on the block chain, from the source of raw materials to the end consumer. This enables complete traceability, making it easier to identify the origin of contaminated or unsafe products.

Smart Contracts: Smart contracts are self-executing contracts with predefined rules and conditions. In a food tracking system, smart contracts can automatically enforce agreements between different parties, such as suppliers, distributors, and retailers. For example, payment can be released to a supplier only when certain conditions are met, such as the successful delivery of safe and high-quality products.

Transparency: Block chain offers transparency into the food supply chain. Consumers can access information about the origins of products, quality control processes, and certifications, allowing them to make more informed choices.

Real-time Data: Block chain allows for real-time data sharing and updates, which means that stakeholders can access the most up-to-date information about the location and condition of food products.

7.2 Feature 2:

Digital Identities: Each food product, packaging, or pallet can have a digital identity on the block chain. This digital identity contains essential information such as batch numbers, production dates, and quality control data.

Sensor Data Integration: Block chain systems can be integrated with IOT (Internet of Things) devices and sensors to monitor temperature, humidity, and other environmental conditions. These sensors can trigger alerts and update the block chain in case of deviations from ideal storage conditions.

Certification and Compliance: Certifications, such as organic, non-GMO, or fair trade, can be recorded on the block chain to verify product claims. Compliance with food safety regulations can also be tracked and verified in real-time.

Secure Data Sharing: Block chain technology allows data sharing among stakeholders in a secure and controlled manner. Participants in the supply chain can access only the data relevant to their role and permissions.

Provenance Tracking: Provenance information, including details about the origin, processing, and handling of food products, is recorded on the block chain, providing confidence in the authenticity and quality of products.

Recall Management: In case of food recalls or safety issues, block chain enables faster and more precise identification of affected products, reducing the scope and impact of recalls.

Decentralization: A decentralized block chain network ensures that no single entity has full control over the system. This enhances trust among stakeholders and reduces the risk of data manipulation.

8. Performance Testing:

8.1Performance Metrics:

Transaction Throughput:

Measure the number of transactions (such as food product updates, verifications, and data entries) the block chain can process per second. High throughput is essential for real-time updates and scalability.

Consensus Algorithm Efficiency:

Evaluate the efficiency of the consensus algorithm used in the block chain (e.g., Proof of Work, Proof of Stake). Calculate the time it takes to confirm transactions and add them to the block chain.

Latency:

Monitor the time it takes for a transaction to be validated and added to the block chain. Low latency is crucial for timely updates and tracking.

Scalability:

Assess the block chain's ability to handle an increasing number of transactions and participants while maintaining performance. Consider vertical and horizontal scalability options.

Block chain Size and Storage:

Track the growth of the block chain's size and storage requirements. Managing data storage is crucial for long-term scalability and performance.

Network Traffic:

Measure the volume of data and network traffic generated by the block chain transactions. High network traffic can affect performance.

Node Synchronization:

Ensure that all nodes in the network are synchronized with the latest block chain data in a reasonable time frame.

Data Consistency:

Verify that data stored on the block chain is consistent and that there are no discrepancies among nodes in the network.

Security and Privacy:

Assess the security and privacy features, such as encryption and access control, which may impact the performance.

Transaction Confirmation Time:

Measure the time it takes for transactions to be confirmed and validated, which can affect the user experience.

Fault Tolerance:

Evaluate the block chain's ability to handle network failures, data corruption, or attacks without compromising performance.

Smart Contract Efficiency:

If using smart contracts, monitor the execution time and gas costs associated with smart contract operations.

Block chain Consensus Nodes:

Keep track of the number of consensus nodes and their geographical distribution to ensure the network's reliability.

Block chain Forks and Reorganizations:

Measure the occurrence of forks and block chain reorganizations, as they can impact data consistency and performance.

Audit and Compliance:

Ensure that the block chain system can meet audit and compliance requirements without significantly impacting performance.

User Experience:

Collect user feedback and satisfaction scores to gauge the overall user experience of the food tracking system.

Energy Consumption:

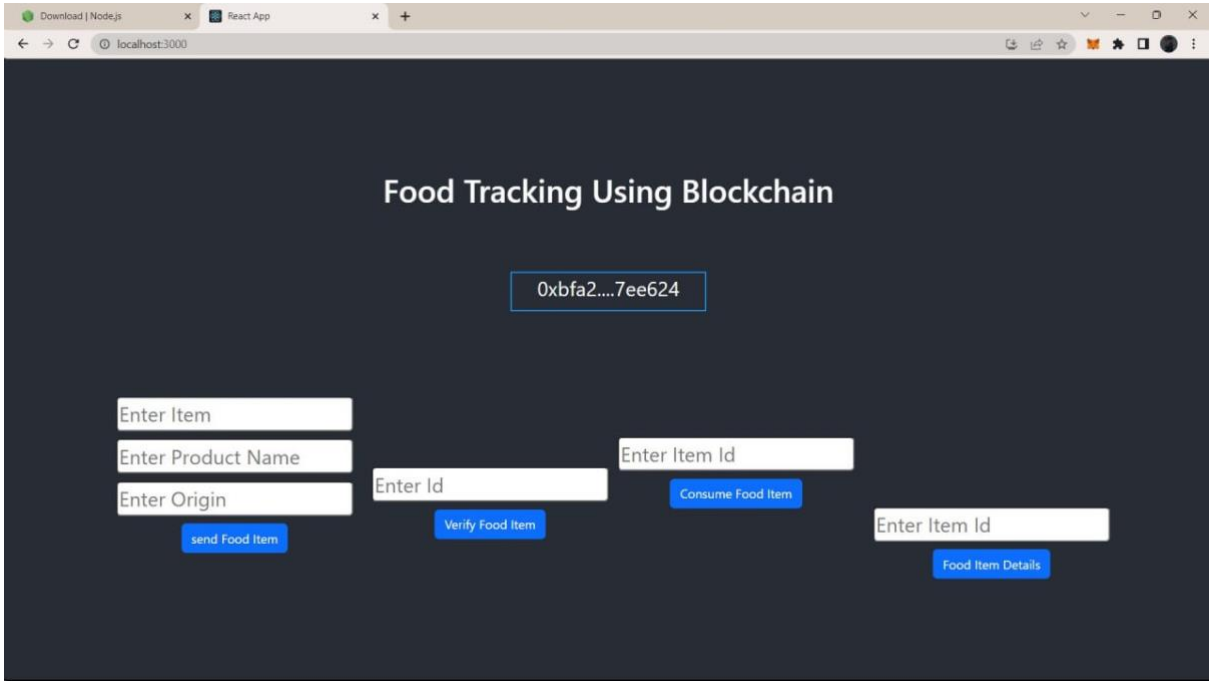
Assess the energy consumption and environmental impact of the block chain network, particularly if using energy-intensive consensus algorithms.

Resource Optimization:

Identify areas where block chain resources can be optimized for cost-efficiency.

9. Results:

9.1Output Screenshot:



10. Advantages & Disadvantages:

Advantages

Transparency and Traceability:

Immutable Records: Block chain creates an immutable ledger of transactions, making it nearly impossible to alter or manipulate data. This ensures that the information about food products is accurate and trustworthy

Food Safety:

Rapid Recall: In the event of a food safety issue or contamination, block chain enables quicker and more precise recalls by identifying the exact source of the problem. This can save lives and reduce economic losses.

Reduced Fraud and Counterfeiting:

Anti-Counterfeiting: Block chain's transparency and security features make it harder for counterfeit products to enter the supply chain. Each product can be assigned a unique digital identity.

Improved Supply Chain Efficiency:

Smart Contracts: Block chain can automate various supply chain processes through smart contracts, which execute actions automatically when predefined conditions are met. This can streamline payments, reduce paperwork, and improve overall efficiency.

Disadvantages:

Complexity and Technical Expertise:

Implementation Complexity: Setting up and maintaining a block chain-based food tracking system can be complex and may require technical expertise. Many food industry stakeholders may lack the necessary skills or resources.

Cost:

High Initial Investment: Developing and implementing a block chain system can be costly, particularly for small and medium-sized businesses. Costs include technology infrastructure, software development, and ongoing maintenance.

Scalability:

Transaction Speed and Capacity: Some block chain platforms struggle with processing a high volume of transactions quickly. In industries with rapid supply chain movements, this can be a significant drawback.

Interoperability:

Lack of Standardization: Different block chain platforms and networks may not be interoperable, making it challenging for various parties in the supply chain to work together. Achieving consensus on standards can be difficult.

Data Privacy:

Over-Transparency: While block chain ensures transparency, there may be sensitive business data and confidential information that stakeholders don't want to share with everyone in the network. Striking the right balance between transparency and privacy is a challenge.

Adoption Barriers:

Resistance to Change: Existing participants in the food supply chain may be resistant to adopting new technology, especially if it disrupts established processes and practices.

Infrastructure:

Technology Access: In some regions, especially in developing countries, access to the necessary technology infrastructure for block chain implementation may be limited.

Regulatory Challenges:

Compliance: Adhering to data protection regulations and other industry-specific regulations while using block chain can be complex and may require legal and regulatory adjustments.

Smart Contracts:

Errors and Bugs: Smart contracts, while automating many supply chain processes, are not immune to coding errors or vulnerabilities, which can have far-reaching consequences.

Energy Consumption:

Some block chain platforms, particularly those using Proof of Work (PoW) consensus mechanisms, consume significant amounts of energy. This environmental impact has raised concerns about sustainability.

Long-Term Viability:

Rapid Technological Evolution: The technology landscape evolves quickly, and block chain systems must evolve to remain effective and secure. An outdated block chain system may become a liability.

User Experience:

User-Friendly Interfaces: The user interfaces of block chain systems can be intimidating and confusing for non-technical users, potentially leading to errors and inefficiencies.

Single Points of Failure:

While block chain is often considered secure, the surrounding infrastructure, such as wallets and access points, can be vulnerable to attacks, potentially resulting in security breaches.

11. Conclusion

In conclusion, the implementation of a food tracking system using block chain technology offers substantial benefits and potential for improving the transparency, security, and efficiency of the food supply chain. It has the power to enhance food safety, reduce fraud, streamline processes, and increase consumer confidence. However, there are notable challenges and disadvantages, including complexity, cost, scalability issues, privacy concerns, and regulatory hurdles.

Successful adoption of block chain in the food industry requires careful planning, collaboration among stakeholders, and ongoing innovation to address these challenges. It's not a one-size-fits-all solution, and the suitability of block chain for a food tracking system depends on the specific needs and constraints of the industry. Despite the disadvantages, block chain technology has the potential to revolutionize how we track, verify, and ensure the quality and safety of the food we consume, promoting a more transparent and trustworthy food supply chain. To realize this potential, stakeholders must work together to overcome the obstacles and build a more secure and efficient food tracking ecosystem.

12.Future scope

Widespread Adoption: As block chain technology matures and becomes more user-friendly, we can anticipate broader adoption across the food supply chain. This will involve more producers, distributors, retailers, and consumers participating in block chain-based systems.

Interoperability and Standards: Efforts are underway to establish industry standards and ensure interoperability between various block chain networks. This will make it easier for different stakeholders and systems to work together seamlessly.

Integration with IoT: The Internet of Things (IoT) can play a crucial role in enhancing food tracking. Combining block chain with IoT devices will enable real-time monitoring and data collection, improving transparency and traceability.

Smart Contracts: Smart contracts will continue to automate various processes within the supply chain, such as triggering payments, quality checks, and routing products based on predefined conditions. This will lead to greater efficiency and cost savings.

Improved Scalability: Block chain platforms are actively working on scalability solutions to handle a higher volume of transactions without compromising speed. This will be essential for large-scale food supply chains.

Enhanced Data Analytics: The data collected through block chain systems will be analyzed to provide insights into supply chain performance, product quality, and consumer behavior. This data-driven decision-making will help optimize operations.

Enhanced Food Safety: Block chain will facilitate faster and more precise food recalls in the event of contamination or safety issues. This will help safeguard public health and reduce economic losses.

Consumer Empowerment: Consumers will have more information about the products they purchase, such as the origin, production methods, and sustainability practices. Block chain will enable them to make informed and ethical choices.

Sustainability and Ethical Sourcing: The ability to track and verify sustainable and ethical practices in the supply chain will become increasingly important. Block chain will assist in promoting transparency and accountability in this regard.

Global Supply Chain Optimization: Block chain can help streamline international trade, reduce bureaucracy, and ensure compliance with varying regulatory requirements. This will benefit global food supply chains.

Reduced Food Waste: Enhanced traceability will lead to better management of perishable products, reducing food waste. Smart contracts can also help automate actions to prevent waste.

Block chain and Tokenization: The integration of block chain with tokenization can enable more efficient supply chain financing, making it easier for small producers to access capital and improve their operations.

Regulatory Compliance: Block chain systems will evolve to ensure better compliance with data protection and other relevant regulations, offering a more secure and trustworthy environment for handling sensitive data.

Fraud Detection and Prevention: The transparency of block chain will make it more challenging for counterfeit products to enter the supply chain, helping to detect and prevent fraud

13.Appendix

Source Code:

// SPDX-License-Identifier: MIT

```
pragma solidity ^0.8.0;

contract FoodTracking {
    address public owner;

    enum FoodStatus {
        Unverified,
        Verified,
        Consumed
    }

    struct FoodItem {
        string itemId;
        string productName;
        string origin;
        uint256 sentTimestamp;
        FoodStatus status;
    }

    mapping(string => FoodItem) public foodItems;

    event FoodItemSent(
        string indexed itemId,
        string productName,
        string origin,
        uint256 sentTimestamp
    );
    event FoodItemVerified(string indexed itemId);
    event FoodItemConsumed(string indexed itemId);

    constructor() {
```

```
    owner = msg.sender;
}

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

modifier onlyUnconsumed(string memory itemId) {
    require(
        foodItems[itemId].status == FoodStatus.Verified,
        "Item is not verified or already consumed"
    );
    _;
}

function sendFoodItem(
    string memory itemId,
    string memory productName,
    string memory origin
) external onlyOwner {
    require(
        bytes(foodItems[itemId].itemId).length == 0,
        "Item already exists"
    );

    foodItems[itemId] = FoodItem({
        itemId: itemId,
        productName: productName,
        origin: origin,
        sentTimestamp: block.timestamp,
```

```
        status: FoodStatus.Unverified
    });

    emit FoodItemSent(itemId, productName, origin, block.timestamp);
}

function verifyFoodItem(string memory itemId) external onlyOwner {
    require(
        bytes(foodItems[itemId].itemId).length > 0,
        "Item does not exist"
    );
    require(
        foodItems[itemId].status == FoodStatus.Unverified,
        "Item is already verified or consumed"
    );

    foodItems[itemId].status = FoodStatus.Verified;

    emit FoodItemVerified(itemId);
}

function consumeFoodItem(
    string memory itemId
) external onlyUnconsumed(itemId) {
    foodItems[itemId].status = FoodStatus.Consumed;

    emit FoodItemConsumed(itemId);
}

function getFoodItemDetails(
    string memory itemId
```

```
    )
        external
        view
        returns (string memory, string memory, uint256, FoodStatus)
    {
        FoodItem memory item = foodItems[itemId];
        return (item.productName, item.origin, item.sentTimestamp, item.status);
    }
}
```

Github:

https://github.com/sanjayoff29

Demo link:

click here