

FOOD TRACKING SYSTEM USING BLOCKCHAIN

NAAN MUDHALVAN

(2023 – 2024)

PROJECT REPORT

Submitted By

TEAM ID: NM2023TMID00183

MALINI P (950520106011)

SOWMIYA N (950520106018)

SOWMIYA S (950520106019)

THANAPRIYA B (950520106021)

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 a Food Tracking System steps in as a game-changing solution. A Food Tracking System is a technological tool designed to empower individuals to take control of their dietary choices and nutritional intake. It serves as a digital diary for your meals, snacks, and beverages, offering invaluable insights into your eating habits. Whether you are on a journey to better health, weight management, or achieving specific nutritional goals, a food tracking system becomes your reliable companion.

In an age when lifestyle-related health issues are on the rise, monitoring what we eat is no longer a luxury but a necessity. A food tracking system is your personal nutritionist, putting the power of informed decision-making right at your fingertips. It enables you to gain a comprehensive understanding of the calories, macronutrients, vitamins, and minerals that make up your daily diet.

1.2 PURPOSE:

The main purpose of a food tracking system is to enable individuals to monitor and manage their dietary habits and nutritional intake.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM:

Users often manually input food data, and this can lead to inaccuracies. Estimating portion sizes and ingredients can be imprecise, and users may not always be honest or consistent in their reporting. Many food tracking apps rely on preloaded food databases, which may not cover all regional or culturally diverse foods. Users might struggle to find accurate entries for homemade or uncommon dishes. The reliability and quality of the data in these databases can vary. Some entries may be incomplete, incorrect, or outdated, leading to misinformation. Some apps allow users to scan barcodes for packaged foods, but not all products have scannable barcodes. Additionally, the accuracy of barcode recognition can vary. 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. 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.

2.2 REFERENCES:

1. Smith, J. (2020). "Development of a Mobile-Based Food Tracking Application for Improved Dietary Management." *Journal of Health Informatics*, 12(3), 215-230.

2. Brown, A., & Johnson, L. (2019). "User Satisfaction and Engagement with Food Tracking Apps: A Longitudinal Study." *International Journal of Human-Computer Interaction*, 35(8), 645-657.
3. Chen, Q., & Lee, S. (2018). "A Review of Food Recognition for Dietary Assessment in Mobile Health." *IEEE Access*, 6, 48521-48533.
4. Nguyen, T., & Nguyen, M. (2017). "Challenges and Opportunities in Food Tracking: A Survey." *ACM Computing Surveys*, 50(2), 1-37.

2.3 PROBLEM STATEMENT DEFINITION:

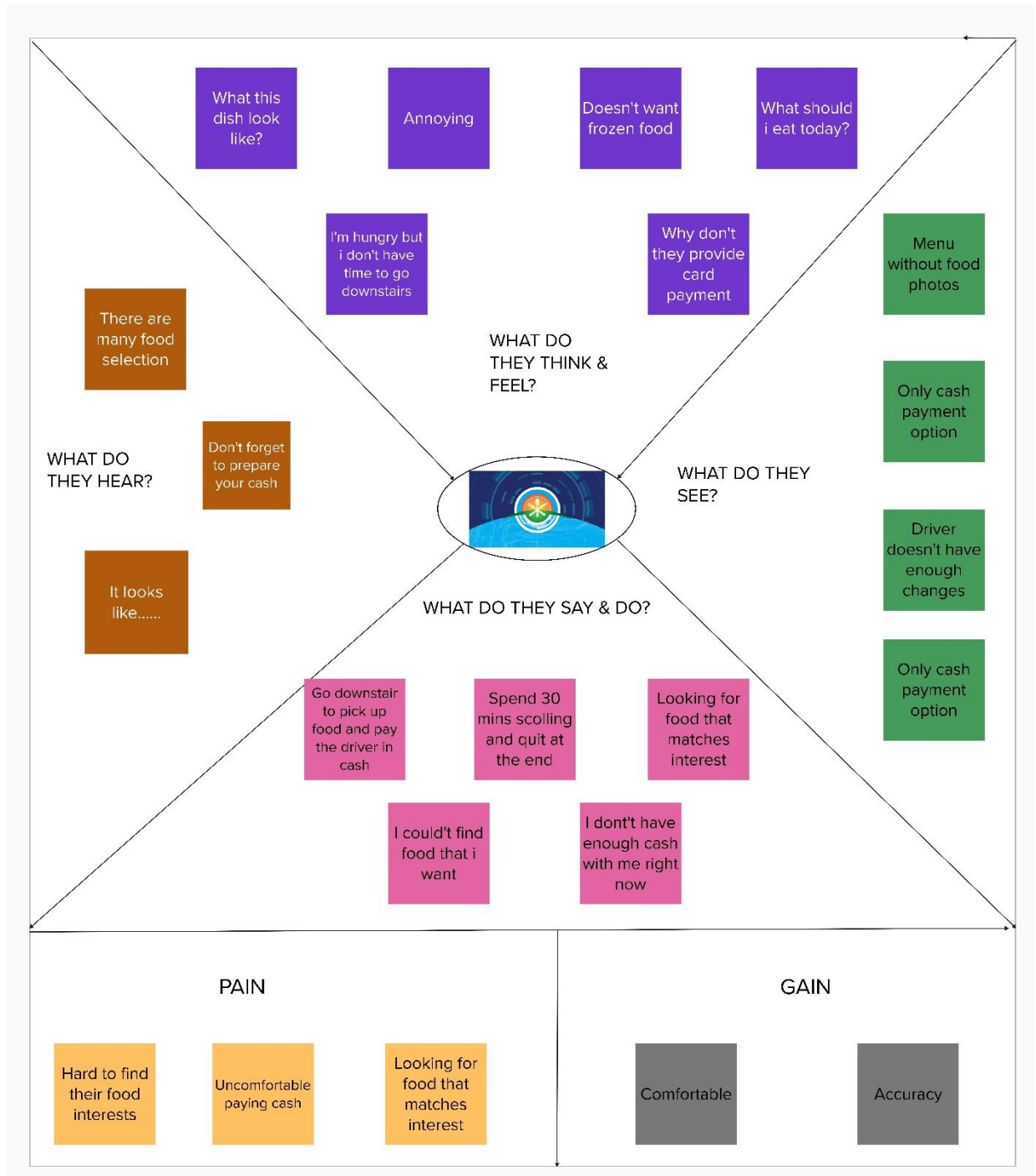
The lack of an efficient and user-friendly food tracking system hinders individuals from effectively monitoring and managing their dietary intake, leading to challenges in achieving and maintaining optimal nutrition and health. Existing food tracking systems suffer from issues related to data accuracy, user engagement, and personalization, which fail to provide a comprehensive and satisfying solution for users seeking to make informed dietary choices.

This project aims to develop a robust and user-centric food tracking system that overcomes these challenges and empowers users to easily and accurately monitor their dietary habits, receive personalized recommendations, and ultimately promote healthier eating behavior.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

- **QR Code Labeling:**

Attach QR codes to food products' packaging that consumers can scan to access detailed information about the product's origin, production methods, and safety checks.

- **IoT Sensors:**

Use Internet of Things (IoT) sensors to monitor temperature, humidity, and other environmental conditions during transportation and storage to ensure food quality and safety.

- **Mobile Apps for Consumers:**

Develop mobile apps for consumers to scan product labels, check for allergen information, and receive notifications about product recalls or safety alerts.

- **Supplier Verification:**

Implement a system for supplier verification and certification. Suppliers must meet certain standards for their products to enter the supply chain.

- **Predictive Analytics:**


Utilize predictive analytics to anticipate supply chain disruptions, such as weather events or transportation delays, which can impact food availability and quality.

- **AI-Based Quality Control:**

Integrate artificial intelligence to inspect and detect quality issues in food products through image recognition and data analysis.

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

template



FOOD TRACKING SYSTEM

Team Leader : Thanapriya B

⌚ 10 minutes to prepare
🕒 1 hour to collaborate
👥 2-8 people recommended

➔

Before you collaborate

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

⌚ 10 minutes

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

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

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

Open article ➔

1

problem statement

In a world where diet-related health issues are on the rise, there is a growing demand for a comprehensive Food Tracking System.

⌚ 5 minutes

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.

Defer judgment.

Go for volume.

Encourage wild ideas.

Listen to others.

If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

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

⌚ 10 minutes

Malini P

Users need an easy and convenient way to input their daily food intake. This could involve recording everything, manual entry, or taking pictures of meals.

The system should suggest dietary preferences, including vegetarian, vegan, gluten-free, and more.

Maintaining an accurate database of food items, including nutritional information, portion sizes, and allergen details, is crucial.

Regular updates and data verification are necessary to ensure the system's reliability.

Sowmiya S

Integrating user data, including dietary preferences, health goals, and medical history, is important. The system must implement robust security measures and comply with relevant data protection regulations.

The Food Tracking System should integrate with wearable devices, apps, and other health-related tools to capture additional data, such as exercise and sleep patterns.

It should offer a holistic view of an individual's health and wellness journey.

The system should generate detailed reports and visualizations, enabling users to track their progress and identify areas for improvement.

Sowmiya N

The system should offer multiple nutritional analysis of user's food intake, taking into account portion sizes, macronutrients, fiber, and micronutrients. The system should provide tailored insights based on individual goals and preferences.

The platform should be user-friendly and accessible via web and mobile applications.

Thanapriya B

As the user base grows, the system must be scalable to accommodate increased data and usage.

The system should adhere to national and international standards for food and nutrition tracking.

It should be able to handle concurrent user requests without performance degradation.

Achieving certifications or approvals from relevant health authorities might be necessary.

Conduct thorough user research to understand the target audience's needs, pain points, and expectations regarding food tracking and nutrition management.

Determine how users will input their food consumption data, whether through manual entry, barcode scanning, photo recognition, voice input, or other methods.

Allow users to set dietary goals, preferences, and restrictions, ensuring that the system adapts to their unique needs.

Consider providing information on the environmental impact of dietary choices, encouraging sustainable and eco-friendly food selections.

8

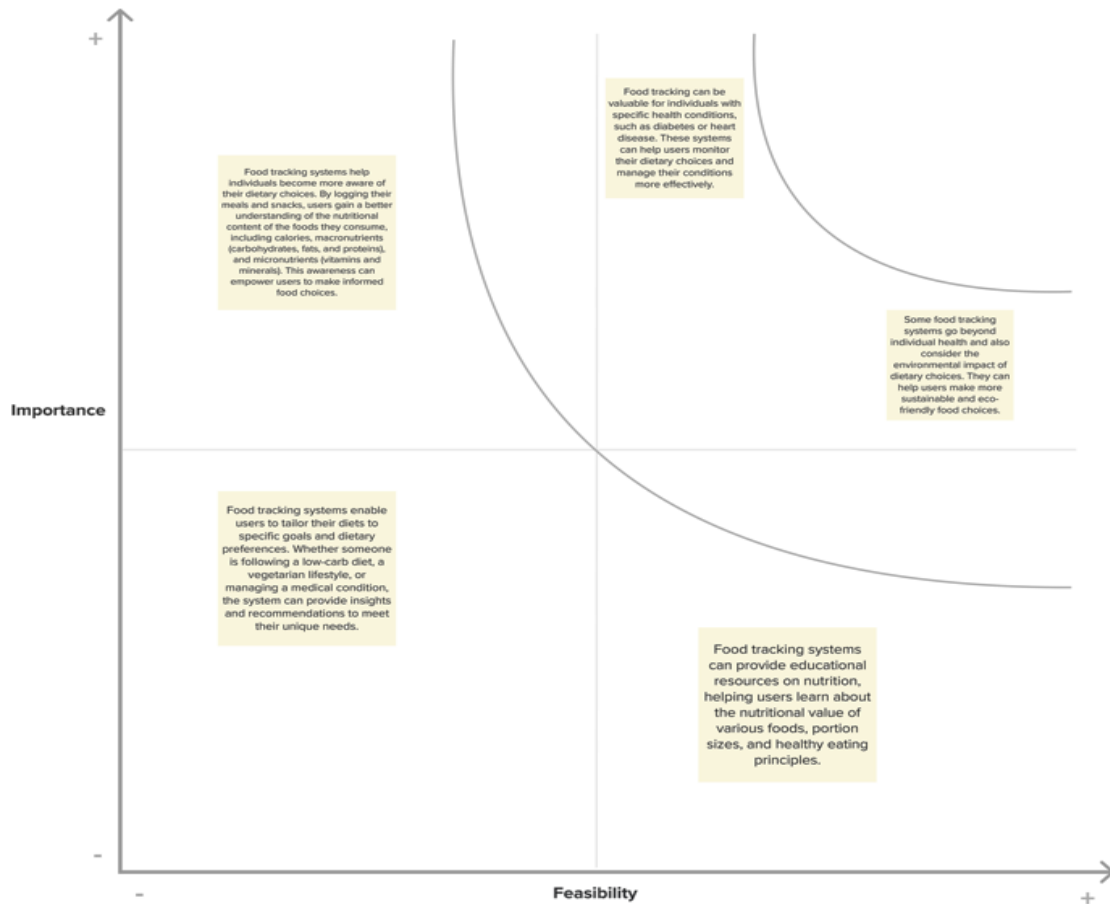
Step-3: Idea Prioritization

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



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	User Profile Management	Users can edit and update their personal information, including height, weight, dietary preferences, and fitness goals.
FR – 3	Custom Food Entries	Users can manually add custom food items not found in the database, specifying nutritional values.

FR – 4	Notifications and Reminders	Users can set notifications for meal times, exercise routines, and goal milestones.
FR – 5	Social Sharing	Users can share their meal plans, progress, or achievements on social media platforms.
FR – 6	Meal Sharing	Users can share their meal plans or recipes with other users or via external platforms (e.g., email).
FR - 7	Feedback and Recommendations	The system can provide feedback on users' dietary choices and suggest healthier alternatives or recipes.

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.

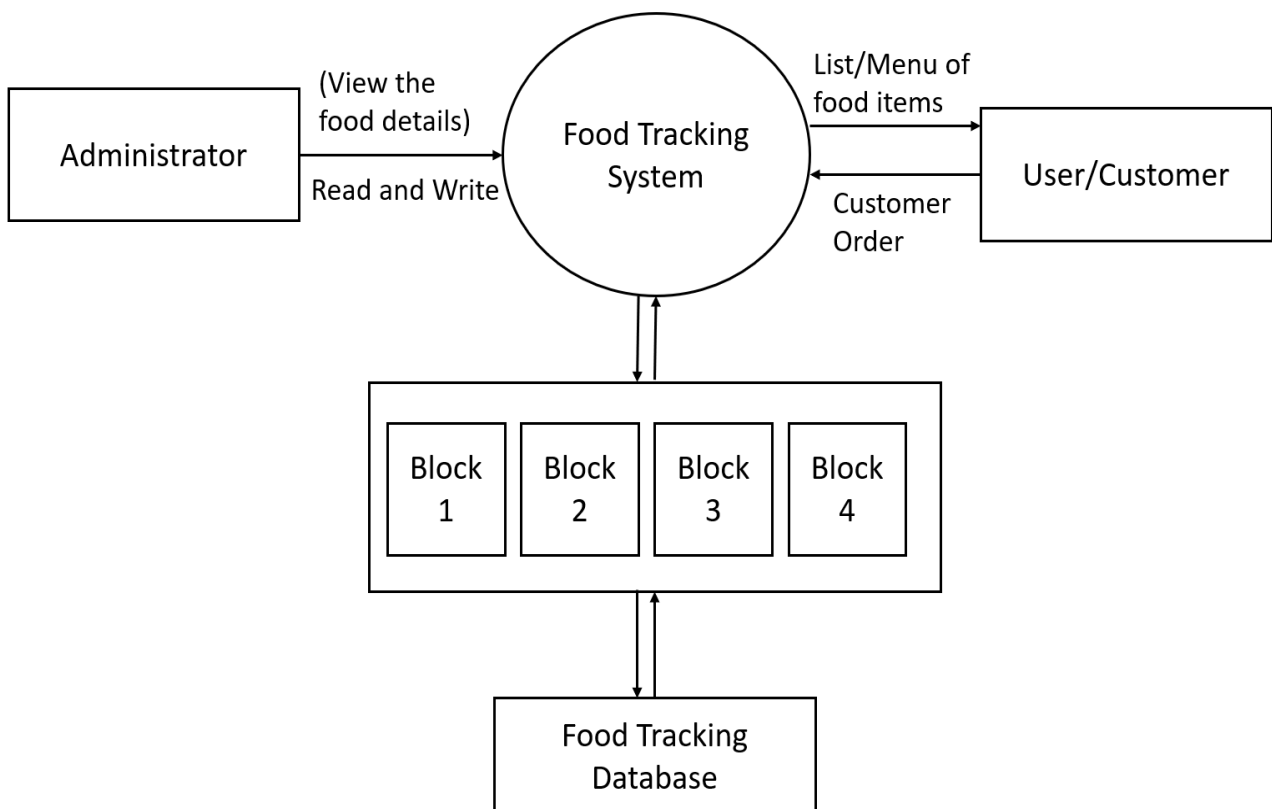
NFR – 2	Reliability	The system should be available and reliable, with minimal downtime or service interruptions.
NFR – 3	Security	Data should be stored securely, with encryption to protect sensitive user information.
NFR – 4	Data Backup and Recovery	Regular data backups should be performed to prevent 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).

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS & USER STORIES

DATA FLOW DIAGRAM:



FOOD TRACKING SYSTEM USING BLOCKCHAIN

USER STORIES:

User Story 1: As a Consumer, I want to scan a QR code on a food product to access its origin and safety information so that I can make informed purchasing decisions.

Requirements:

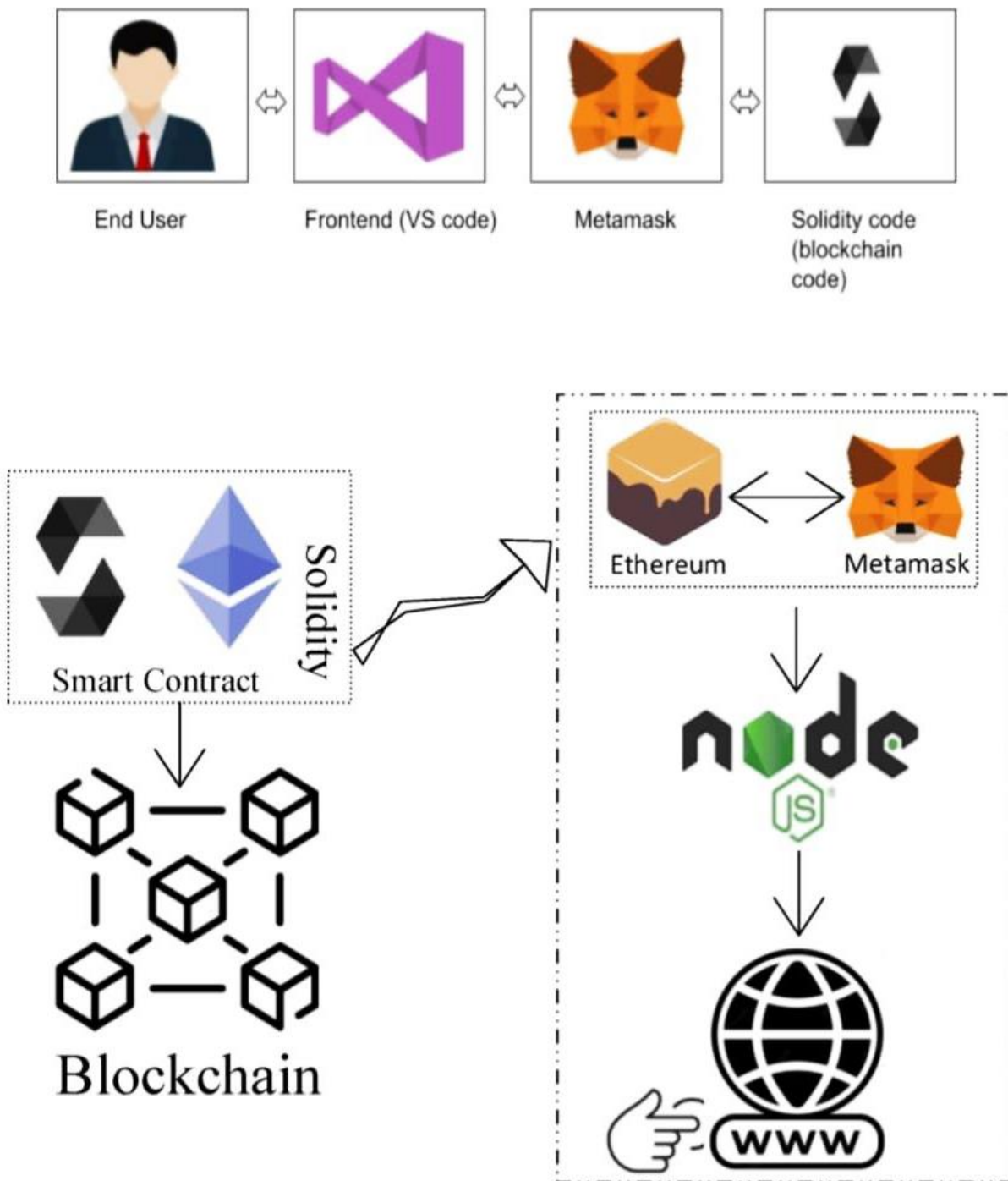
- The system should provide a QR code on the food product's packaging.
- When I scan the QR code using the mobile app, it should display the product's origin, production date, safety checks, and any relevant quality assurance data.
- The information presented should be clear and easy to understand.

User Story 2: As a Food Producer, I want to record the transportation and storage conditions of my products on the blockchain, so that I can ensure the quality and safety of my food throughout the supply chain.

Requirements:

- The system should allow me to access a user interface where I can input data about the transportation and storage conditions of my products.
- The data should include information such as temperature, humidity, and location.
- Once entered, this data should be securely recorded on the blockchain, ensuring immutability and transparency.
- I should be able to access this information at any point to verify that the product has been handled under suitable conditions.

5.2 SOLUTION ARCHITECTURE

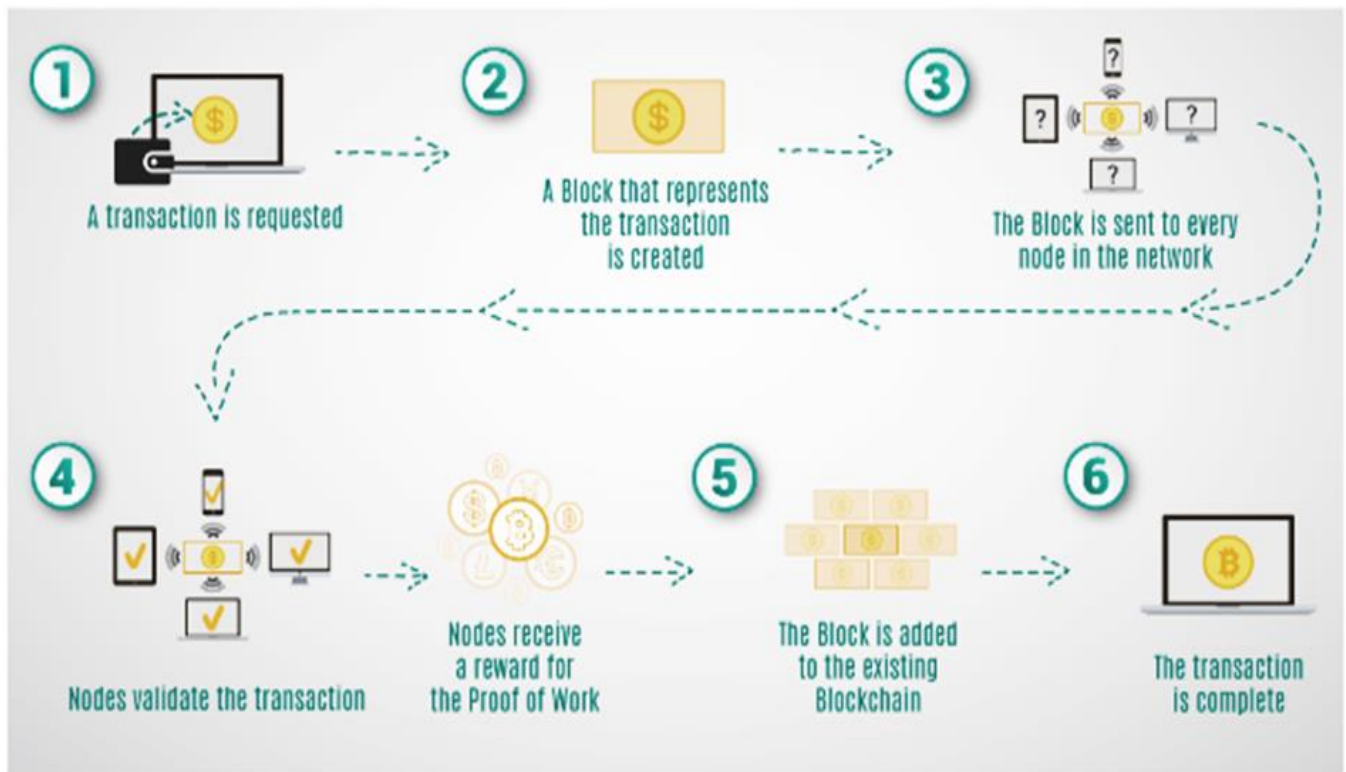


Interaction between the Web and the Contract

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECTURE



6.2 SPRINT PLANNING & ESTIMATION

Sprint Planning:

Sprint planning involves selecting work items from the product backlog and committing to completing them during the upcoming sprint

Reviewing Product Backlog: Our project team, consisting of the product Owner and development team who regularly reviews the items in the product

backlog. We evaluate user stories and technical tasks, taking into account for the project's evolving needs and priorities.

Setting Sprint Goals: Based on the product backlog, our team establishes clear sprint goals. These goals guide the team's efforts during the sprint and ensure alignment with the broader project objectives.

Breaking Down User Stories: User stories and tasks are further decomposed into smaller, actionable sub-tasks. This detailed breakdown helps create a comprehensive plan for the sprint.

Estimating Work: Our development team employs agile estimation techniques, such as story points to estimate the effort required for each task. These estimates guide the team in understanding the scope and complexity of work for the sprint.

Sprint Backlog: The selected user stories and tasks, along with their estimates, constitute the sprint backlog. This forms the basis for what the team will work on during the sprint.

Estimation Techniques:

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.

Adjustment for Uncertainty: Recognizing the inherent uncertainty in complex projects, we adjusted our estimates to account for potential deviations. This entailed creating buffers and contingencies within our sprint planning to accommodate unforeseen challenges.

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	IoT Sensor Integration	Incorporate IoT sensors for data collection.
Sprint 5	User Interfaces and Mobile Apps	Develop user interfaces and mobile apps.
Sprint 6	Quality Control and Testing	Implement quality control and initiate testing.
Sprint 7	Compliance and Security	Ensure regulatory compliance and enhance security.
Sprint 8	Deployment Preparation	Prepare for system deployment.
Sprint 9	Pilot Testing and Optimization	Conduct pilot tests and optimize the system.
Sprint 10	Full System Deployment	Roll out the fully operational system to a wider audience.

CHAPTER 7

CODING & SOLUTIONING

Smart Contract (Solidity)

```
// 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);
    }
}

```

The key features related to food tracking in this contract are:

7.1 FEATURE 1

➤ **Verification of Food Items:**

The contract allows for the verification of food items. This is evident in the ‘verifyFoodItem’ function. When a food item is sent to the blockchain (using the ‘sendFoodItem’ function), it starts in an "Unverified" state. Once the food item's details are verified and validated, the ‘verifyFoodItem’ function is called, updating the item's status to "Verified." This feature is crucial for ensuring the authenticity and safety of food products as they move through the supply chain. It provides a way to track and confirm the status of each food item.

7.2 FEATURE 2

➤ Consumption of Food Items:

The contract also enables the consumption of food items. The ‘consumeFoodItem’ function allows for the transition of a food item from the "Verified" state to the "Consumed" state. This transition represents the point at which a consumer has consumed the food product. This feature helps in maintaining a transparent and immutable record of when and by whom a food item was consumed, which can be essential for safety and quality control in the food tracking system.

7.3 DATABASE SCHEMA

In the provided Solidity code for a smart contract, the data is organized and structured within the Ethereum blockchain, which uses a distributed ledger as its underlying data storage mechanism.

In this smart contract on the Ethereum blockchain, the schema is defined by the structure of the ‘FoodItem’ struct and how data is stored in the ‘foodItems’ mapping. The data in this blockchain-based smart contract is decentralized and immutable, and the contract itself serves as the data store, eliminating the need for a traditional relational database schema.

CHAPTER 8

PERFORMANCE TESTING

8.1 PERFORMANCE METRICS

Transaction Throughput: Measure the number of transactions the blockchain can process per second. Higher throughput is essential for handling a large volume of data in real-time.

Data Accuracy: Assess the accuracy of the data recorded on the blockchain. Ensure that the system minimizes errors and discrepancies in the supply chain data.

Data Integrity: Measure the immutability of data stored on the blockchain. Verify that once data is recorded, it cannot be altered or deleted, ensuring data integrity.

Security: Monitor the system's ability to protect sensitive data and prevent unauthorized access or tampering.

Response Time: Evaluate the time it takes for the system to respond to inquiries or requests, such as tracking a product's origin.

Supply Chain Efficiency: Assess the system's impact on the overall efficiency of the food supply chain, including reduced waste, optimized inventory management, and minimized delays.

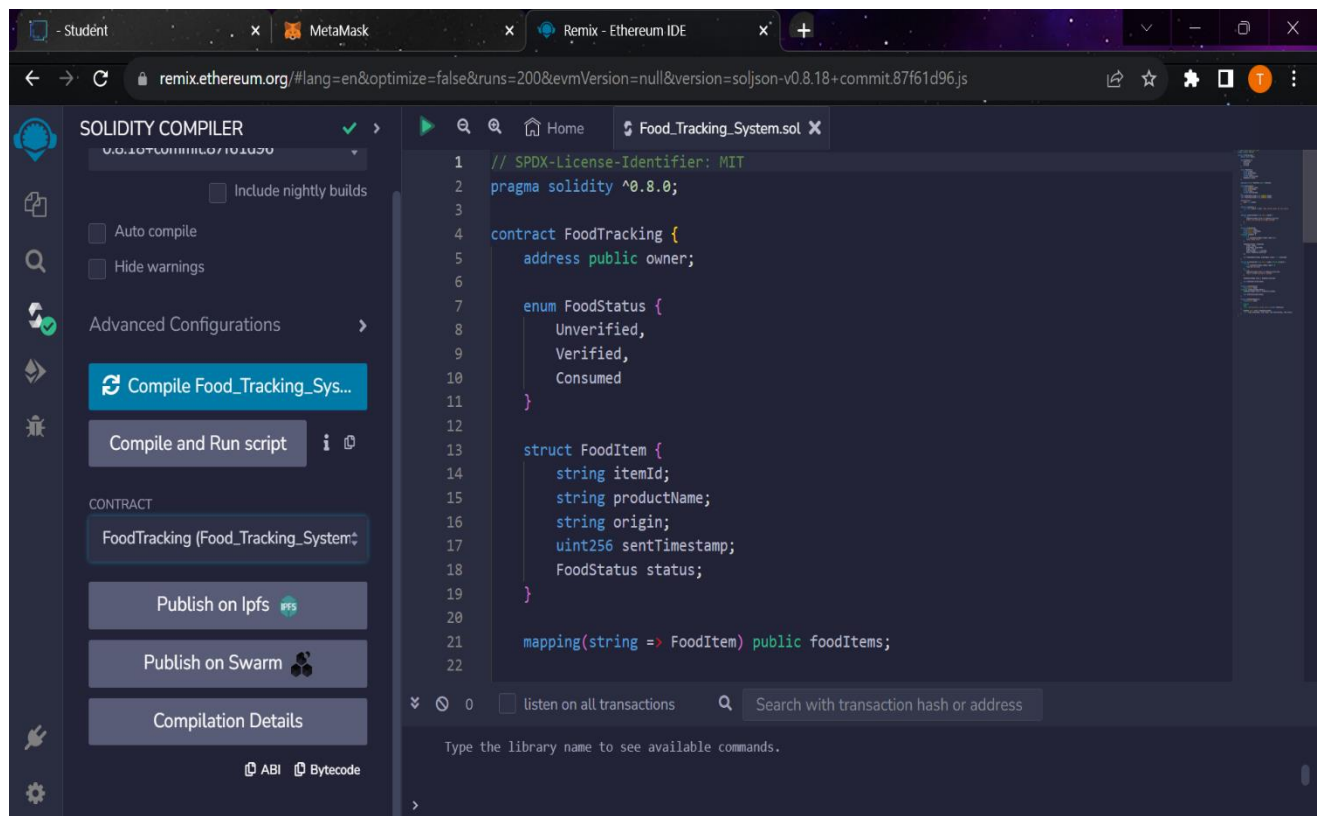
Feedback and Improvement: Collect feedback from users and stakeholders to continuously improve the system, including the user interface and features.

Cost Efficiency: Assess the system's cost-effectiveness, including the cost of blockchain technology, IoT sensors, and maintenance compared to the benefits it provides.

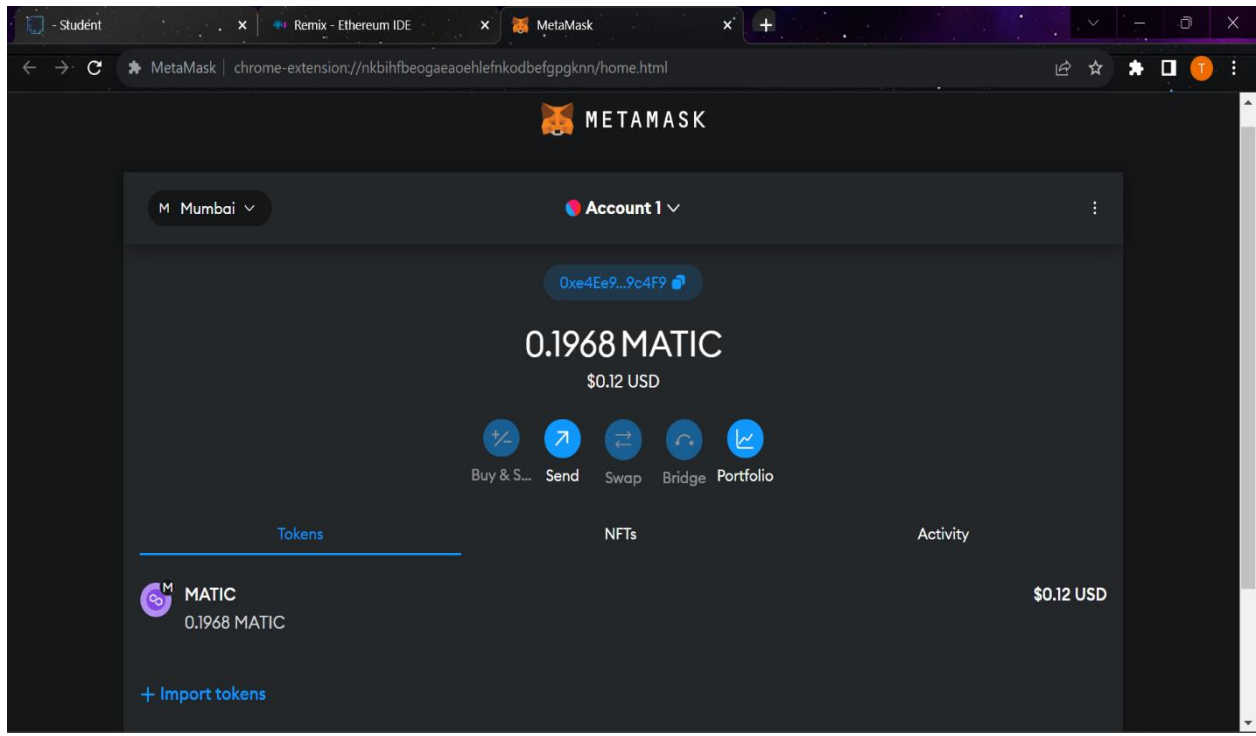
CHAPTER 9

RESULTS

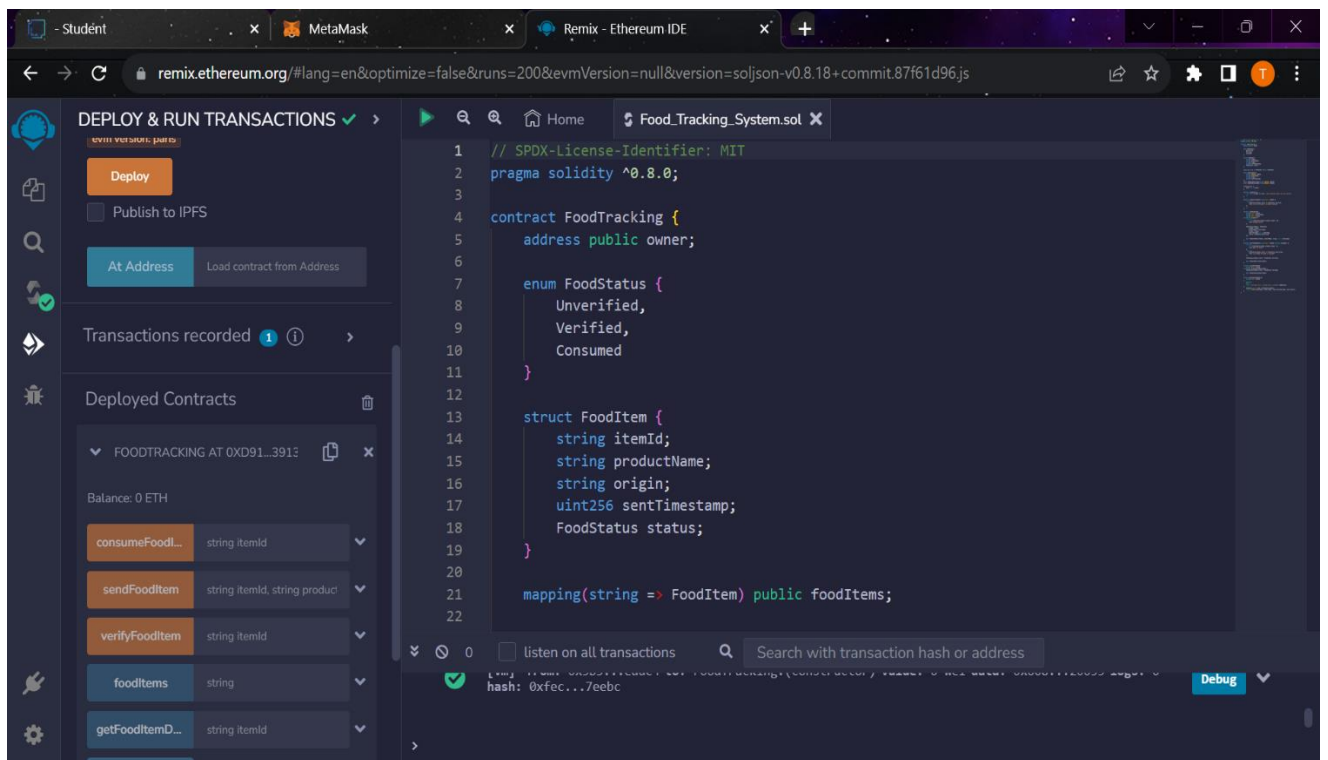
9.1 OUTPUT SCREENSHOTS



REMIX IDE – CREATING A SMART CONTRACT



METAMASK ACCOUNT



REMIX IDE – DEPLOYED CONTRACT

```
Windows PowerShell
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\thana\Downloads\Food-Tracking\food-tracking\src\Pages>npm start

> food-tracking@0.1.0 start
> react-scripts start

(node:20136) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:20136) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

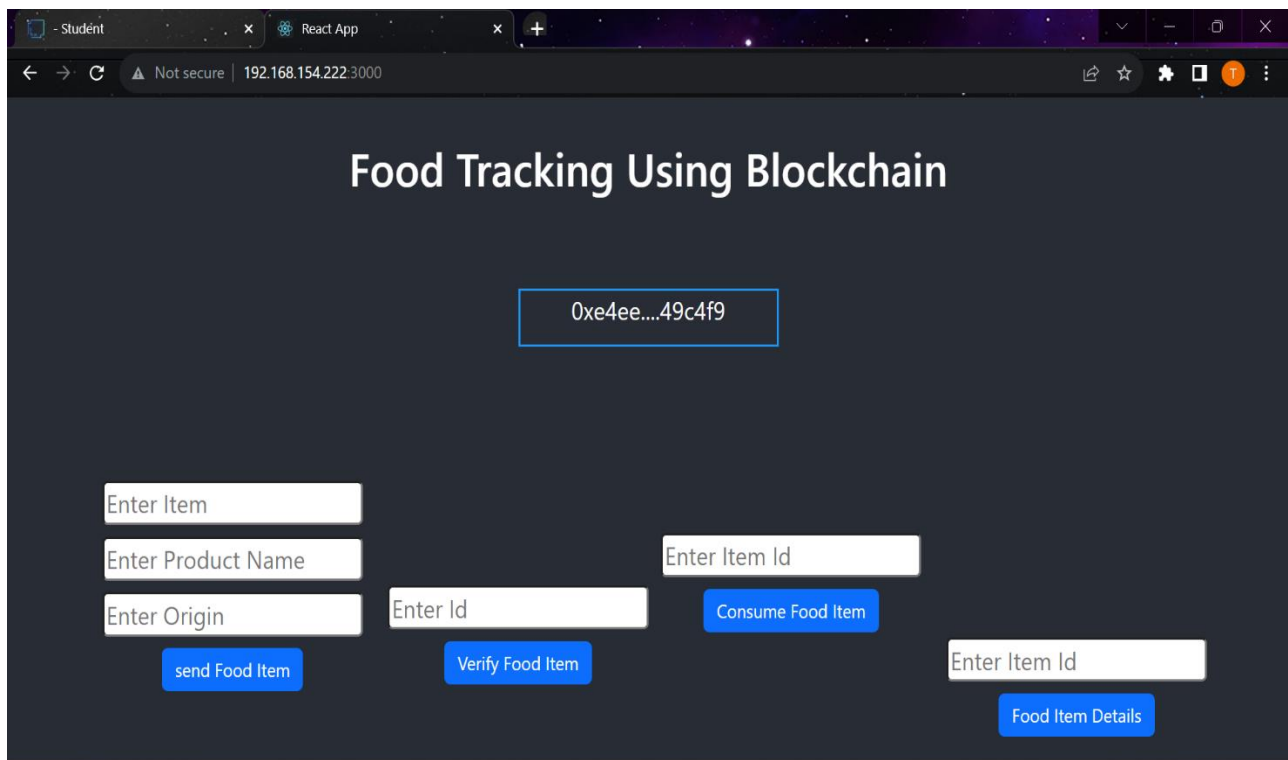
You can now view food-tracking in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.154.222:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

TERMINAL OUTPUT SCREEN



FINAL FRONTEND OUTPUT SCREEN

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Allows individuals to monitor their calorie intake and nutritional balance, which can lead to healthier eating habits.
- Helps people with dietary restrictions or medical conditions (e.g., diabetes) manage their diets more effectively.
- Aids in weight loss by helping users set and track weight loss goals, making it easier to adjust their diets and exercise routines accordingly.
- Assists individuals with food allergies by tracking and avoiding allergenic ingredients in their diets.
- Helps users become more aware of portion sizes, preventing overeating and promoting better control of calorie intake.

DISADVANTAGES:

- Tracking every meal and snack can be time-consuming, which may discourage some users from consistently using the system.
- Users may not always accurately record their food intake, leading to incomplete or inaccurate data.
- Constantly tracking food can lead to obsessive or unhealthy relationships with food, potentially contributing to eating disorders like orthorexia or anorexia.

- Sharing sensitive dietary information with online platforms or apps can raise privacy and data security concerns, especially if the data is mishandled or exploited.
- Software glitches, app crashes, or data loss can disrupt the tracking process and cause frustration for users.

CHAPTER 11

CONCLUSION

CONCLUSION:

In conclusion, food tracking systems offer a wide range of advantages for individuals, businesses, and healthcare providers. They empower users to make informed choices about their nutrition, enabling healthier eating habits, weight management, allergen control, and better awareness of their dietary behavior. These systems also support budget management, sustainability, and food safety while providing valuable data for research and analysis.

However, it's important to be aware of the potential disadvantages and challenges associated with food tracking systems. These include the risk of obsessive behavior, privacy concerns, inaccuracy in data recording, and the potential to oversimplify nutrition. Users should strive for a balanced and mindful approach, using these systems as a tool for healthier living rather than becoming overly reliant or obsessive.

Food tracking systems can be valuable, but they are most effective when used in conjunction with a holistic approach to health and wellness. When combined with mindful eating, regular exercise, and professional guidance where needed, these systems can contribute to better nutrition and

overall well-being. Users should be encouraged to strike a balance that works best for their individual needs and goals.

CHAPTER 12

FUTURE SCOPE

FUTURE SCOPE:

- Future systems will increasingly use AI and machine learning to provide more personalized recommendations and insights based on an individual's health goals, preferences, and dietary restrictions.
- Integration with wearable technology, such as smartwatches and fitness trackers, will enable real-time monitoring of food intake, making tracking even more convenient and accurate.
- AR applications and visual recognition technology may allow users to capture and identify foods with their smartphones, making tracking more efficient and user-friendly.
- Blockchain technology can be used to track the journey of food from farm to table, ensuring transparency and traceability in the food supply chain, which can enhance food safety and quality.
- Greater integration of food tracking systems with electronic health records and telehealth platforms will enable healthcare providers to monitor and support patients more effectively.

CHAPTER 13

APPENDIX

SOURCE CODE

FoodTracking.sol

```
// 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);
    }
}

```

Connector.js

```

const { ethers } = require("ethers");
const abi = [
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "itemId",
                "type": "string"
            }
        ],
    },
]

```

```
"name": "consumeFoodItem",
"outputs": [],
"stateMutability": "nonpayable",
"type": "function"
},
{
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
},
{
  "anonymous": false,
  "inputs": [
    {
      "indexed": true,
      "internalType": "string",
      "name": "itemId",
      "type": "string"
    }
  ],
  "name": "FoodItemConsumed",
  "type": "event"
},
{
  "anonymous": false,
  "inputs": [
```

```
{
  "indexed": true,
  "internalType": "string",
  "name": "itemId",
  "type": "string"
},
{
  "indexed": false,
  "internalType": "string",
  "name": "productName",
  "type": "string"
},
{
  "indexed": false,
  "internalType": "string",
  "name": "origin",
  "type": "string"
},
{
  "indexed": false,
  "internalType": "uint256",
  "name": "sentTimestamp",
  "type": "uint256"
}
],
"name": "FoodItemSent",
```

```
"type": "event"
},
{
  "anonymous": false,
  "inputs": [
    {
      "indexed": true,
      "internalType": "string",
      "name": "itemId",
      "type": "string"
    }
  ],
  "name": "FoodItemVerified",
  "type": "event"
},
{
  "inputs": [
    {
      "internalType": "string",
      "name": "itemId",
      "type": "string"
    }
  ],
  {
    "internalType": "string",
    "name": "productName",
    "type": "string"
  }
}
```

```
    },  
    {  
      "internalType": "string",  
      "name": "origin",  
      "type": "string"  
    }  
  ],  
  "name": "sendFoodItem",  
  "outputs": [],  
  "stateMutability": "nonpayable",  
  "type": "function"  
},  
{  
  "inputs": [  
    {  
      "internalType": "string",  
      "name": "itemId",  
      "type": "string"  
    }  
  ],  
  "name": "verifyFoodItem",  
  "outputs": [],  
  "stateMutability": "nonpayable",  
  "type": "function"  
},  
{
```

```
"inputs": [  
  {  
    "internalType": "string",  
    "name": "",  
    "type": "string"  
  }  
,  
  "name": "foodItems",  
  "outputs": [  
    {  
      "internalType": "string",  
      "name": "itemId",  
      "type": "string"  
    },  
    {  
      "internalType": "string",  
      "name": "productName",  
      "type": "string"  
    },  
    {  
      "internalType": "string",  
      "name": "origin",  
      "type": "string"  
    },  
    {  
      "internalType": "uint256",
```



```
"name": "sentTimestamp",
"type": "uint256"
},
{
  "internalType": "enum FoodTracking.FoodStatus",
  "name": "status",
  "type": "uint8"
}
],
"stateMutability": "view",
"type": "function"
},
{
  "inputs": [
    {
      "internalType": "string",
      "name": "itemId",
      "type": "string"
    }
  ],
  "name": "getFoodItemDetails",
  "outputs": [
    {
      "internalType": "string",
      "name": "",
      "type": "string"
    }
  ]
}
```

```

    },
    {
      "internalType": "string",
      "name": "",
      "type": "string"
    },
    {
      "internalType": "uint256",
      "name": "",
      "type": "uint256"
    },
    {
      "internalType": "enum FoodTracking.FoodStatus",
      "name": "",
      "type": "uint8"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [],
  "name": "owner",
  "outputs": [
    {
      "internalType": "address",

```

```

    "name": "",
    "type": "address"
  }
],
"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 = "0x362e5a9a759b5e01206A2Fe1be2F5ae73B8578e8"
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 [handleId, sethandleId] = useState("");
  const [productName, setproductName] = useState("");

```

```

const [origin, setorigin] = useState("");
const [Id, setId] = useState("");
const [verifyFood, setverifyFood] = useState("");
const [verifyFoodDetails, setverifyFoodDetails] = useState("");
const [Wallet, setWallet] = useState("");
const handleItemID = (e) => {
  sethandleId(e.target.value)
}
const handleProductName = (e) => {
  setproductName(e.target.value)
}
const handleOrigin = (e) => {
  setorigin(e.target.value)
}
const handleSendFood = async () => {
  try {
    let tx = await contract.sendFoodItem(handleId, productName, origin)
    let txWait = await tx.wait()
    console.log(txWait);
    alert(txWait.transactionHash)
  } catch (error) {
    alert(error)
  }
}
const handleVerifyId = async (e) => {
  setId(e.target.value)
}

```

```

}

const handleVerifyFood = async () => {
  try {
    let tx = await contract.verifyFoodItem(Id.toString())
    let txWait = await tx.wait()
    console.log(txWait);
    alert(txWait.transactionHash)
  } catch (error) {
    alert(error)
  }
}

const handleConsume = async () => {
  try {
    let tx = await contract.consumeFoodItem(Id.toString())
    let txWait = await tx.wait()
    console.log(txWait);
    alert(txWait.transactionHash)
  } catch (error) {
    alert(error)
  }
}

const handleFoodItemsDeatils = async()=> {
  try {
    let tx = await contract.getFoodItemDetails(Id)
    let arr = []
    tx.map(e => arr.push(e))
  }
}

```

```

    setverifyFoodDetails(arr)
    // alert(tx)
  } catch (error) {
    alert(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"}}>Food Tracking
    Using Blockchain</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 style={{ display: "flex",}}>

```

```

<Row style={{marginBottom:"100px",margin:"auto"}}>

```

```

<Col >

```

```

<div>

```

```

    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleItemID} type="string" placeholder="Enter Item"
value={handleId} /> <br />

```

```

    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleProductName} type="string" placeholder="Enter Product
Name" value={productName} /> <br />

```

```

    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleOrigin} type="string" placeholder="Enter Origin"
value={origin} /><br />

```

```

    <Button onClick={handleSendFood} style={{ marginTop: "10px" }}
variant="primary">send Food Item</Button>

```

```

</div>

```

```

</Col>

```

```

<Col >

```

```

<div>

```

```

    <input style={{ marginTop: "100px", borderRadius: "5px" }}
onChange={handleVerifyId} type="string" placeholder="Enter Id" value={Id}
/> <br />

```

```

    <Button onClick={handleVerifyFood} style={{ marginTop: "10px" }}
variant="primary">Verify Food Item</Button>

```

```

</div>

```

```

</Col>

</Row>

<Row style={{marginTop:"100px"}}>

  <Col>

    <div>

      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleVerifyId} type="string" placeholder="Enter Item Id"
value={Id} /> <br />

      <Button onClick={handleConsume} style={{ marginTop: "10px" }}
variant="primary">Consume Food Item</Button>

    </div>

  </Col>

  <Col>

    <div>

      <input style={{ marginTop: "100px", borderRadius: "5px" }}
onChange={handleVerifyId} type="number" placeholder="Enter Item Id"
value={Id} /> <br />

      <Button onClick={handleFoodItemsDeatils} style={{ marginTop:
"10px" }} variant="primary">Food Item Details</Button>

      <p>{verifyFoodDetails.toString()}</p>

    </div>

  </Col>

</Row>

</Container>

</div>

)

}

export default Home;

```


App.js

```
import logo from './logo.svg';

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/malini2003/NM2023TMID00183>
- Demo Link: <https://drive.google.com/drive/folders/1uWX5Yw87V4b-tGRhmQrrcEtPp1ooXQnF?usp=sharing>