FOOD TRACKING SYSTEM

Team I'd : NM2023TMID00900

BY,

LOGESH T

ARUNKUMAR E

SELVAVINAYAGAM M

KISHANTH S

FOOD TRACKING SYSTEM

Introduction

PROJECT OVERVIEW:

The food supply chain is extremely complex especially as products move between many players in different countries. The current challenges with traditional methods of food traceability include incomplete or inaccurate recordkeeping, slow response times during recalls, supply chain fraud, lack of data transparency between supply chain stakeholders, and difficulty tracing products through multiple steps in the supply chain.

Blockchain technology is becoming increasingly integrated in the food supply chain to enhance traceability and safety. With blockchain food traceability systems, every step of the journey from farm to consumer can be recorded and easily accessed.

PURPOSE:

Food traceability is important for many reasons including the ability to quickly respond to contamination issues, ensuring product safety, decreasing foodborne illness risk, and verifying claims about a product's ingredients.

Current food supply chain systems are dominantly outdated, have caused many foodborne illness outbreaks around the world, are rampant with supply chain fraud, and often have a lack of data transparency between supply chain stakeholders.

Blockchain food traceability systems have many benefits including helping to effectively contain contamination outbreaks, enabling people to trace a product within seconds instead of weeks, creating an auditable trail of accurate data in a tamper-resistant way, and significantly lowers costs.

With Verifiable Credentials backed by blockchain technology, companies can enable customers to see where the products came from and their details by simply scanning a product QR code with their phone.

S.NO	Title of the paper	Authors	Algorithm	Advantages	Disadvantages
1.	FOOD TRACEABILITY SYSTEM USING BLOCKCHAIN	IUON-CHANG LIN, HSUAN SHIH, JUI-CHUN LIU, YI-XIANG JIE	Sensors are used in an IOT mode Traceability system	Allows for real-time monitoring of food items,improving efficiency and reducing uncertainties.	System failures, or technical glitches.complex supply chain
2.	**Blockchain for Food Tracking**	Arif Furkan Mendi	Gps , sensor	Helps comply with food safety regulations and standards. Reducing waste and optimizing processes.	Requiring continuous updates and adjustments to tracking systems Leading to complexities in its implementation.
3.	Blockchain use cases for food traceability and control	Axfoundation, SKL Kommentus, Swedish county councils and regions, Martin & Servera, and Kairos Future.	a unique identifier such as a barcode, QR code, or a RfID transmitter	Reduce foodborne illnesses,distribution is secure and tamper-resistant.	Lack of access to technology Leading to incomplete or inaccurate data.
4.	Food Supply Chain Traceability using Block Chain	S.Kayalvizhi, D. Amirtha Sughi, G.Shivasree, J.Shruthi	A radio frequency Identification (RFID)-based sensor,MQTT,IOT.	Aiding in weight management, and helping meet nutritional goals.	The data is not adequately protected. Potentially leading to user fatigue.
5.	Applying blockchain technology to improve agrifood traceability	Huanhuan Fenga,b, Xiang Wanga,b, Yanqing Duan c , Jian Zhangd*, Xiaoshuan Zhanga,b*	Internet of Things Wireless Sensor Network (WSN), QR code, NFC,And RFID	Reduce foodborne illnesses, and enable consumers to make more informed choices about the products they consume.	People may not accurately measure or record their food intake, leading to inaccuracies in the nutritional data.

Ideation Phase
Define the Problem Statements

Date	17-10-2023
Team ID	NM2023TMID00900
Project Name	Project – Food Tracking System

Customer Problem Statement Template:

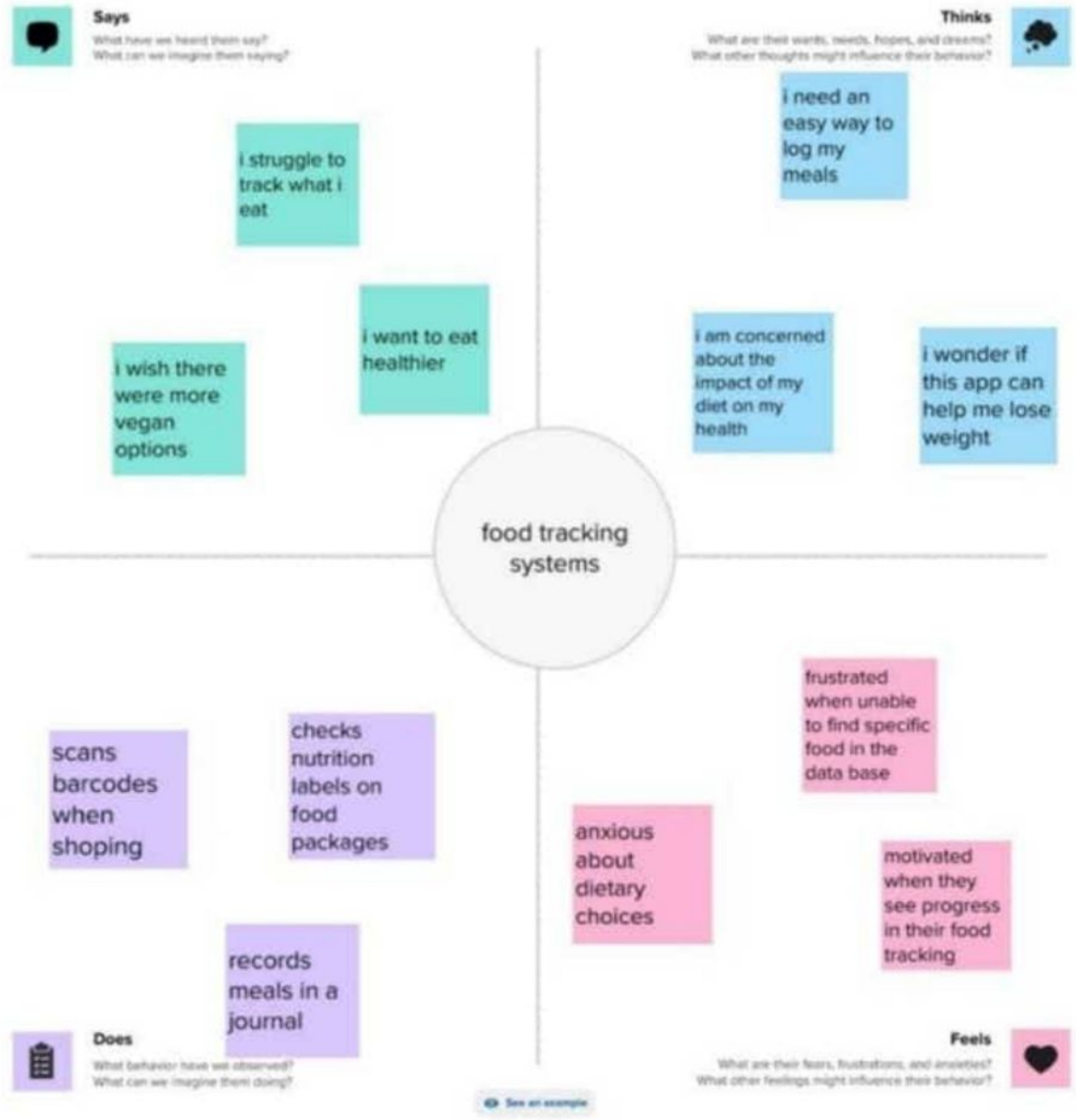
Ideation Phase
Empathize & Discover

Date	20-10-2023
Team ID	NM2023TMID00900
Project Name	FOOD TRACKING SYSTEM

Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users.
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



Ideation Phase
Brainstorm & Idea Prioritization Template

Date	20/10/2023
Team ID	NM2023TMID00900
Project Name	Project-food tracking system

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

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

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

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

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we [your problem statement]?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

- ability to follow the movement of food product
- packaging and distribution process
- widely used in barcode scanning method

Person 2

- monitoring of the manufacturing process
- it involves documenting and linking process
- identifying cost effective meal option

Person 3

- food tracking used in barcode scanning and RFID tags
- helps keep track of food in supply chain
- website or app used in food tracking

Person 4

- keep track of and record the histry of an item
- allow track the status and location
- apps are used in food tracking system

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

apps are used in food tracking

cost effective meal option

allow the track status and location

keep track of food suply chain

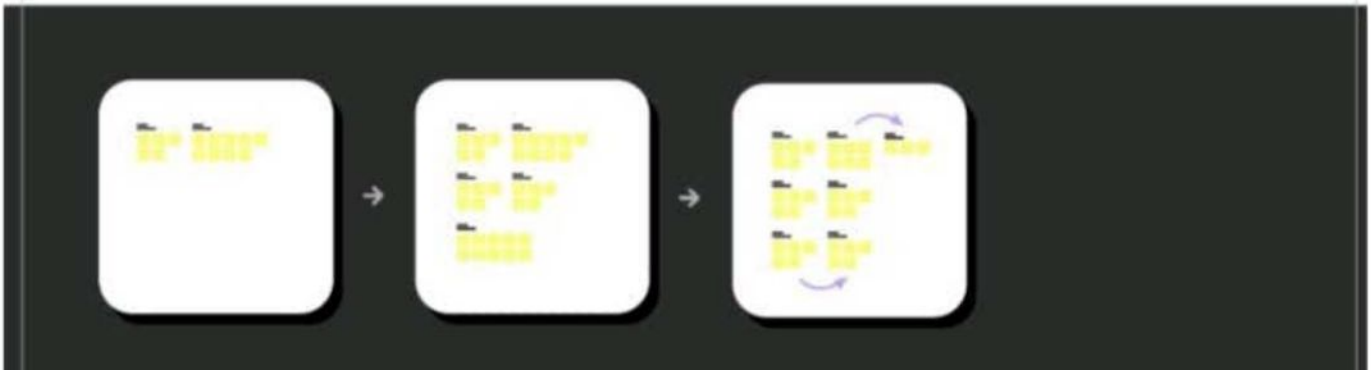
monitoring the manufacturing process

widely used in barcode scanning

ability to follow the movement of food product

involves documentig and linking process

apps are used in food tracking system



4

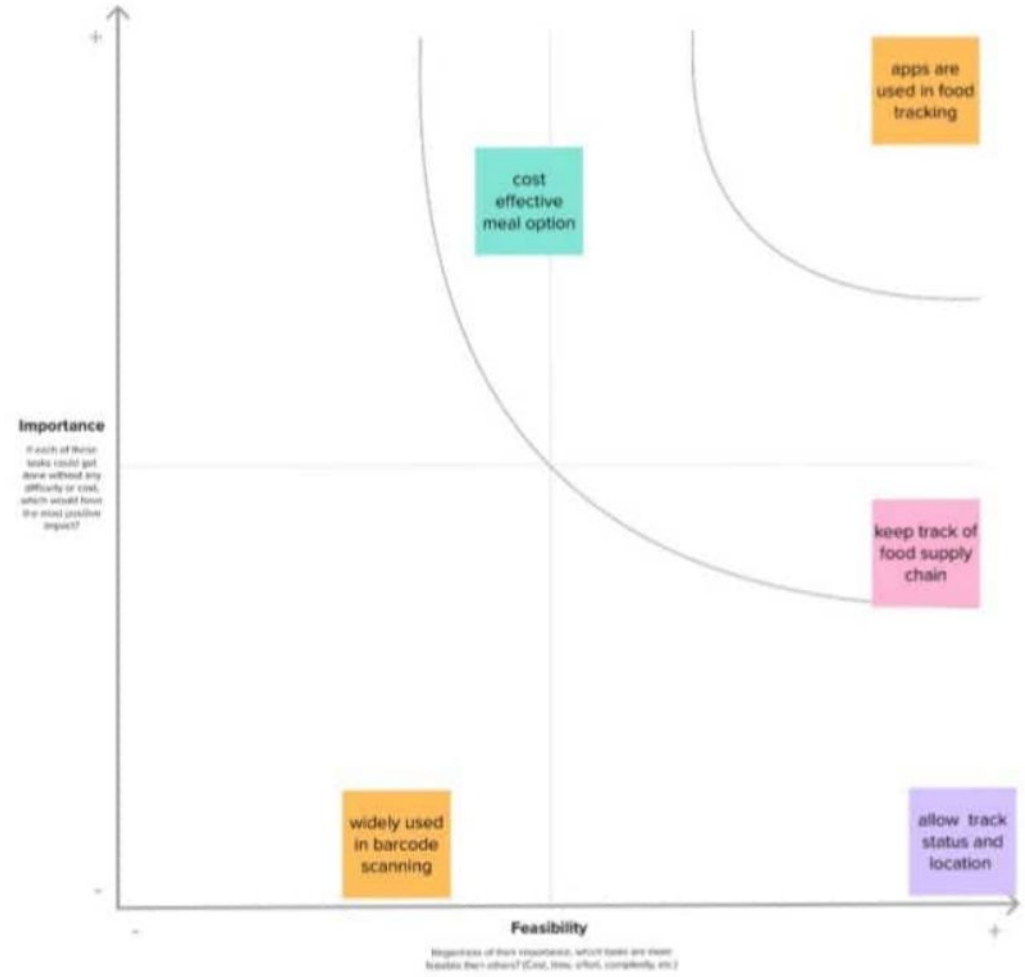
Prioritize

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 cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer hosting the **H key** on the keyboard.



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	20 oct 2023
Team ID	NM2023TMID00900
Project Name	Project –FOOD TRACKING SYSTEM

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Remix IDE	generally refer to the specific dependencies and configurations needed for your smart contract development.
FR-2	Visual code	The condition should be a boolean expression, and the "Error message" will be displayed if the condition evaluates to false.
FR-3	Metamask extension	Metamask is a popular browser extension for managing Ethereum wallets and interacting with decentralized applications

Non-Functional Requirements:

Following are the non-functional requirements of the proposed solution.

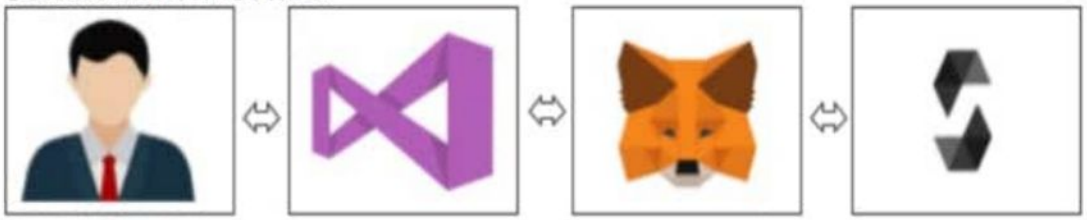
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user interface should be intuitive and user-friendly, ensuring that users of all technical levels can use the system effectively
NFR-2	**Security**	Data must be securely stored and transmitted, and user accounts and information should be protected from unauthorized access
NFR-3	**Reliability**	The system should be available and operational 24/7, with minimal downtime for maintenance
NFR-4	**Performance**	The system should respond quickly to user interactions, with minimal latency when adding, editing, or retrieving food data.

NFR-5	**Availability**	Mobile-based food tracking apps are generally available as long as the user's device is operational. However, the app's availability is contingent on the user's device functioning correctly and having an active internet connection if data synchronization is required.
NFR-6	**Scalability**	The system should be able to handle a growing number of users and data without a significant drop in performance.

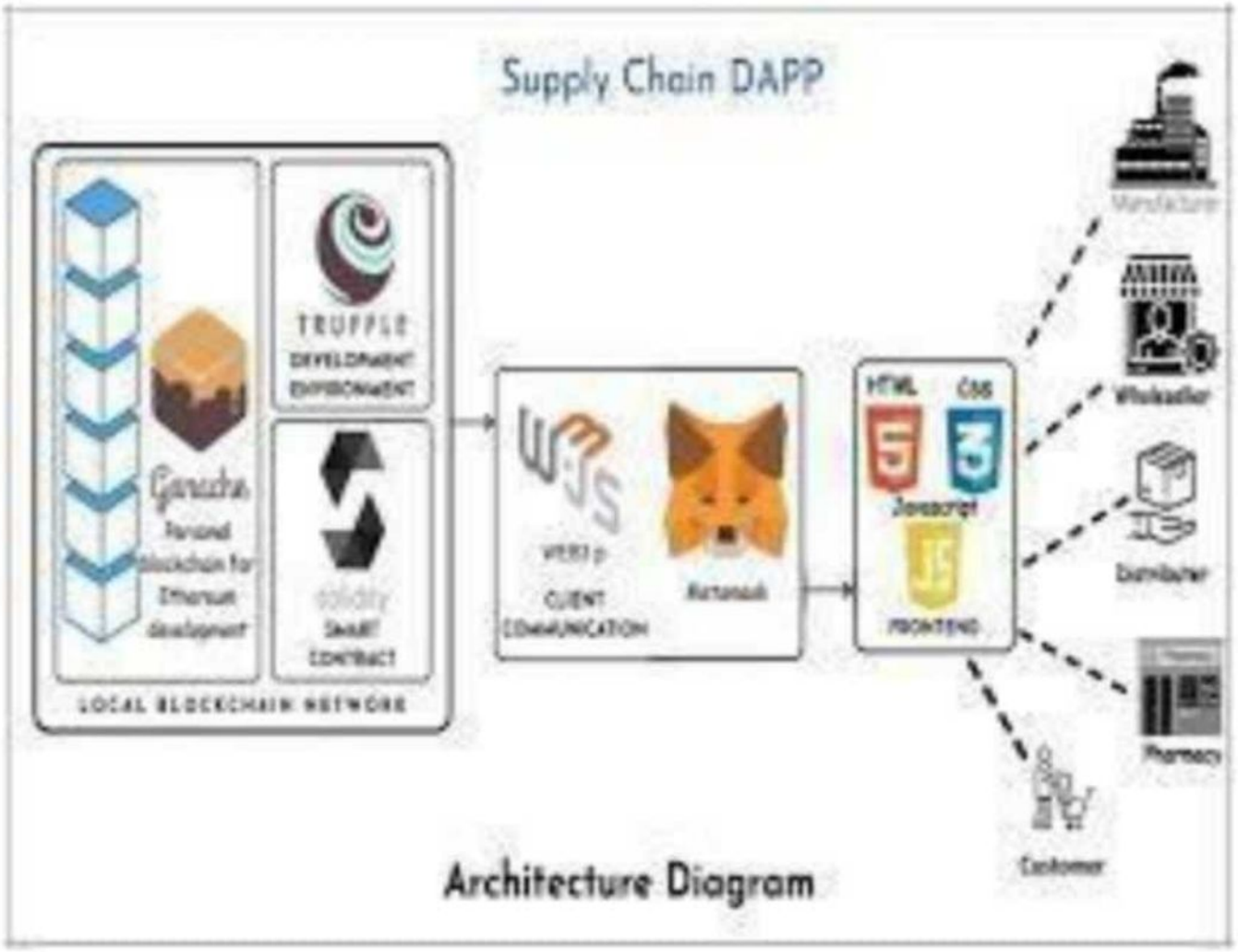
Solution Architecture

Date	20 September 2023
Team ID	NM2023TMID00900
Project Name	Project – FOOD TRACKING SYSTEM
Maximum Marks	

Solution Architecture:



End User | Frontend (VS code) | Metamask | Solidity code (blockchain code)



CODING & SOLUTIONING

REMIX IDE CODE:

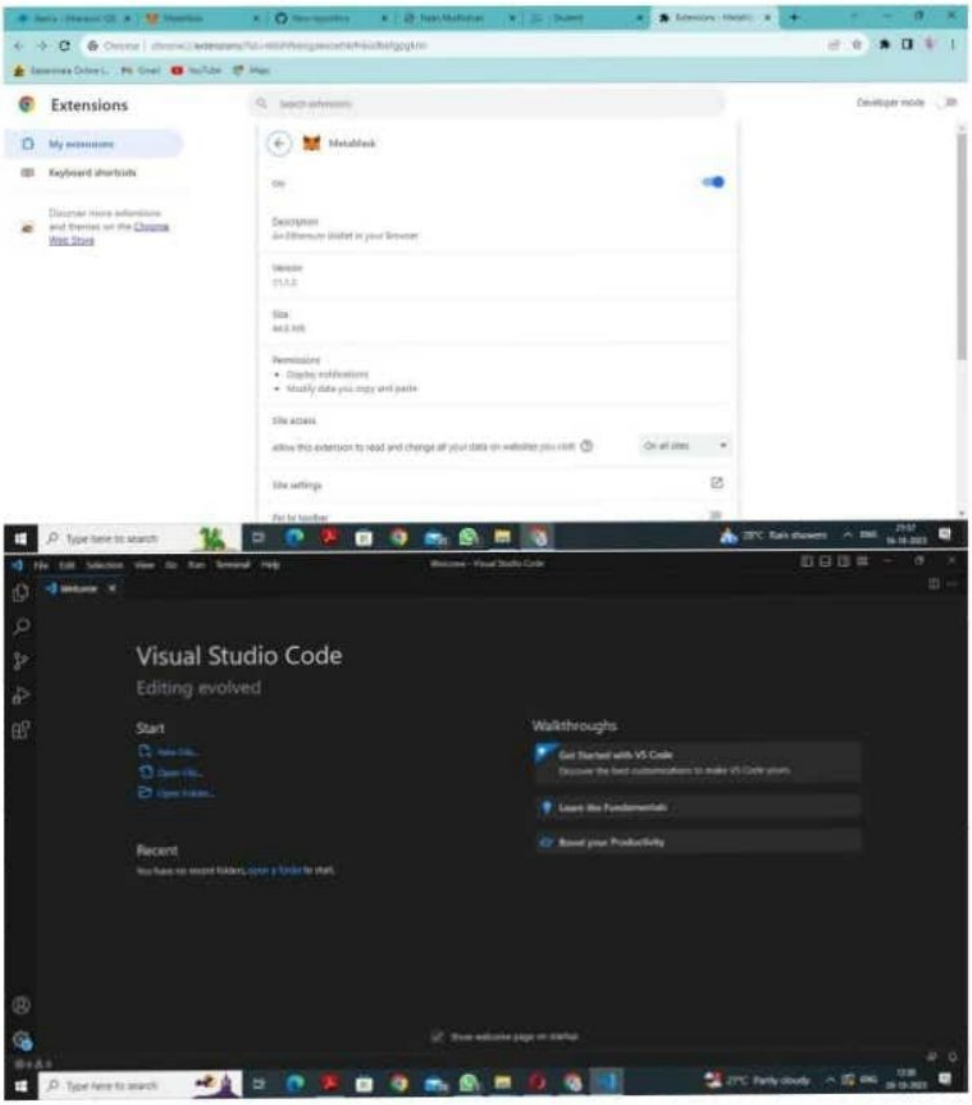
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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,
```

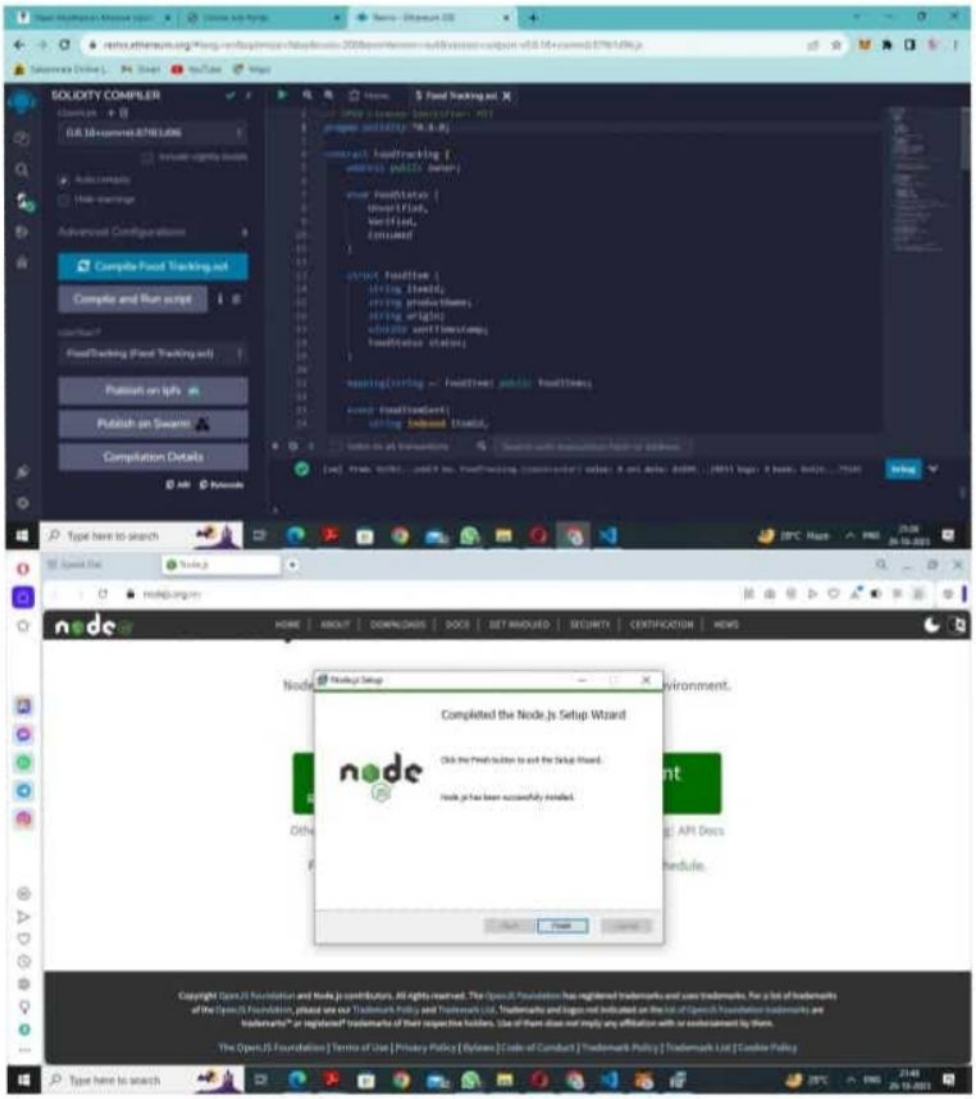
```
    string productName,
    string origin,
    uint256 sentTimestamp
);
event FoodItemVerified(string indexed itemId);
event FoodItemConsumed(string indexed itemId);

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"
    );
```

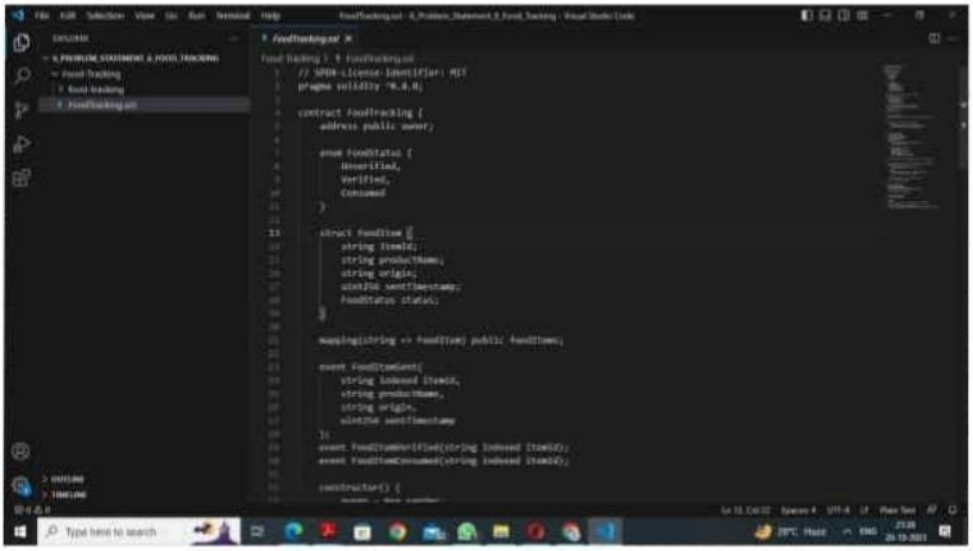
```
        foodItems[itemId].status = FoodStatus.Verified;

        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);
    }
}
```

1.INFORMATION GATHERING:



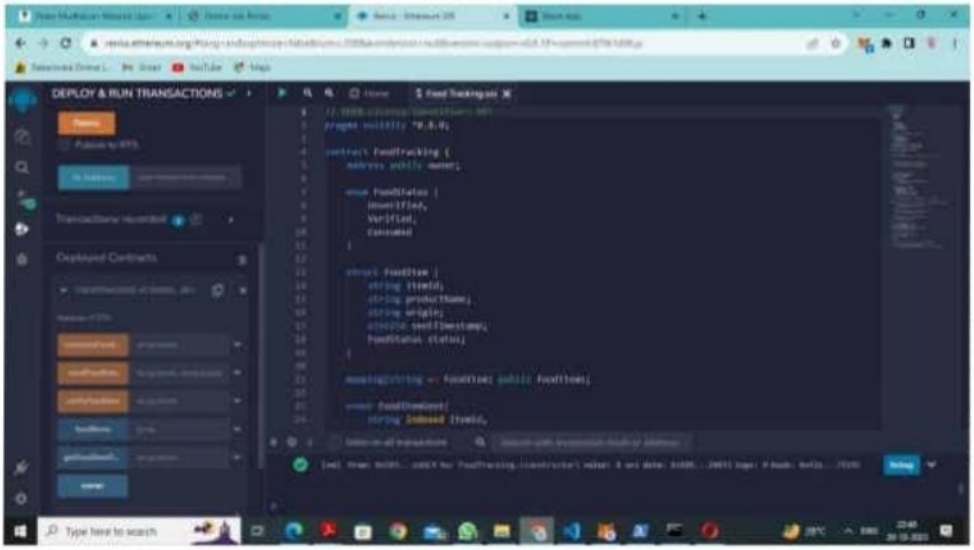


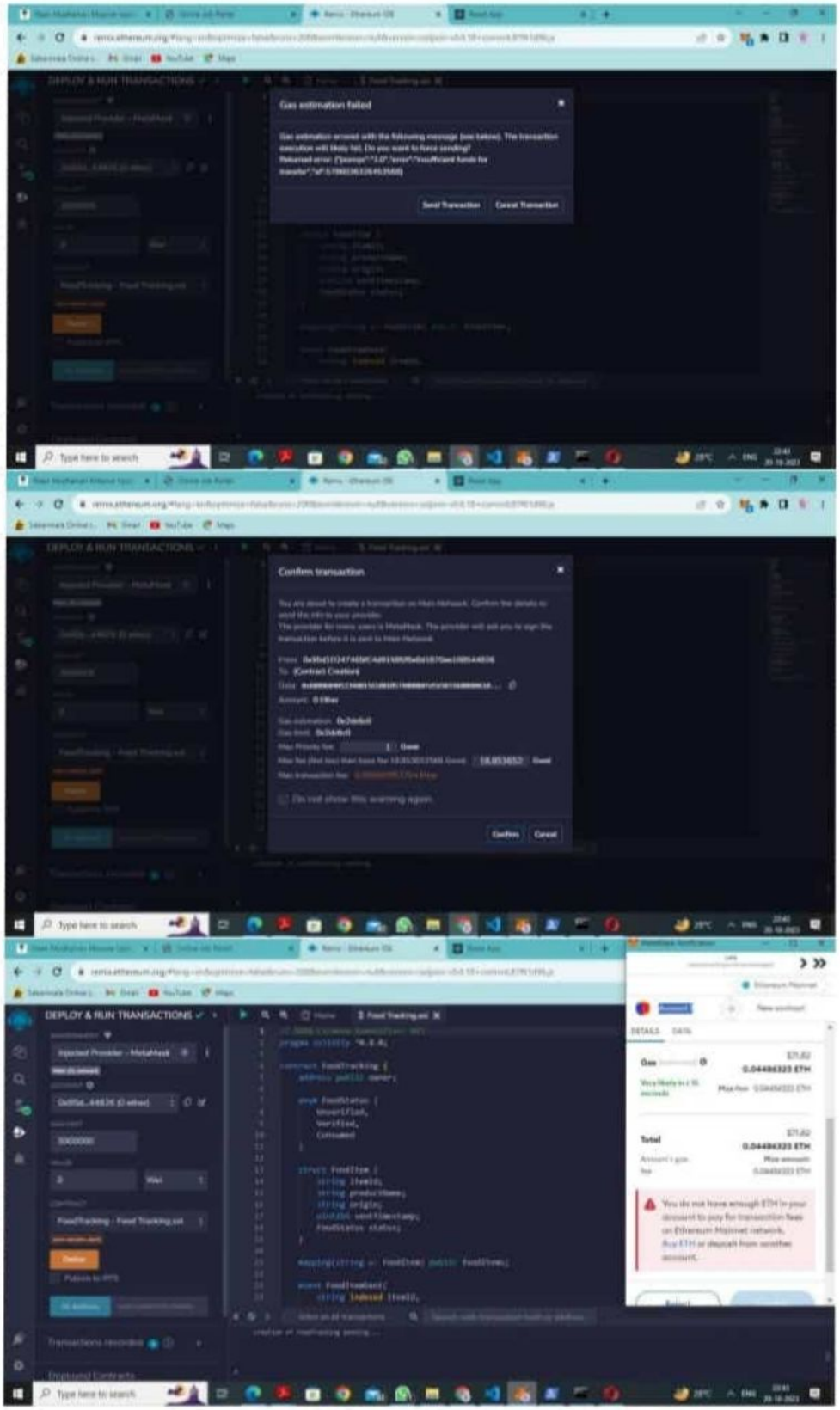
2.EXTRACT THE ZIP FILES:



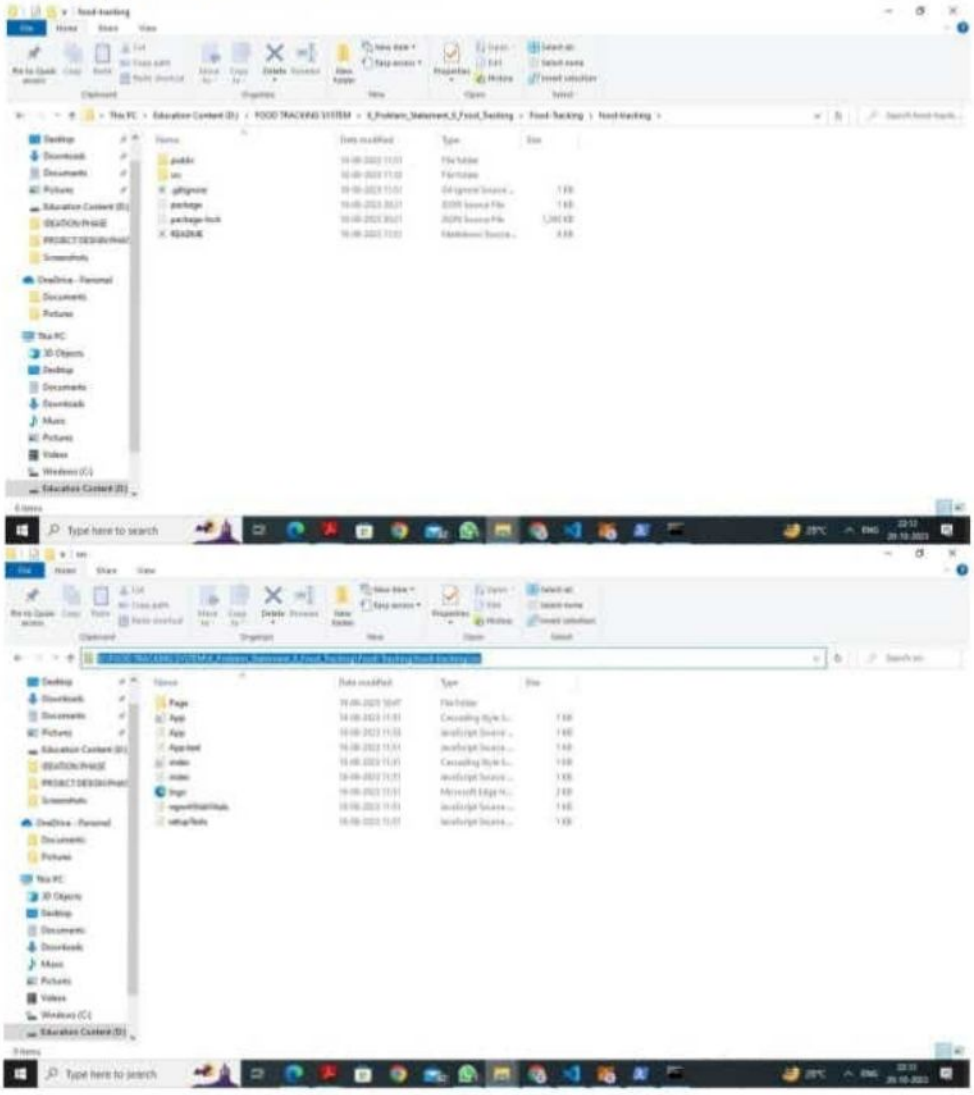
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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() {
```

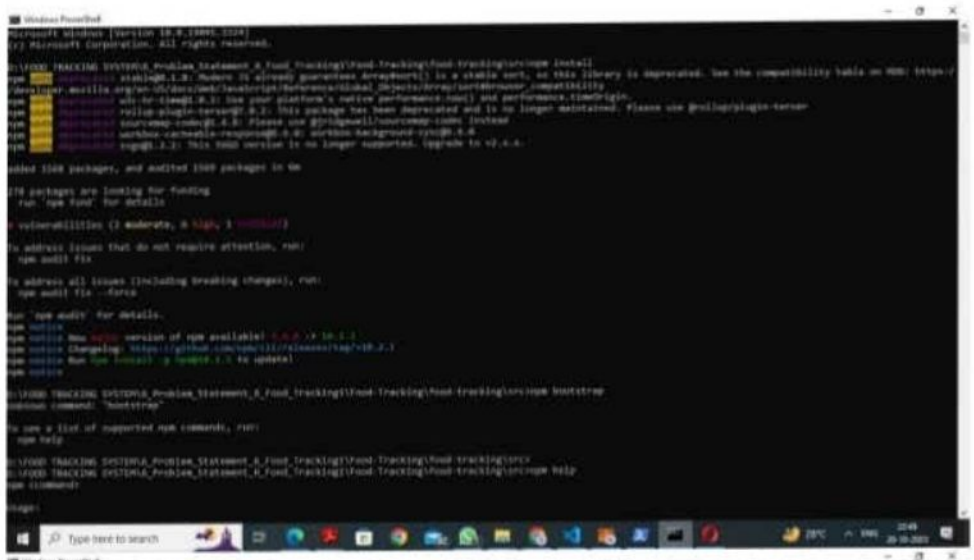
3.REMIX IDE PLATFORM EXPLORING:

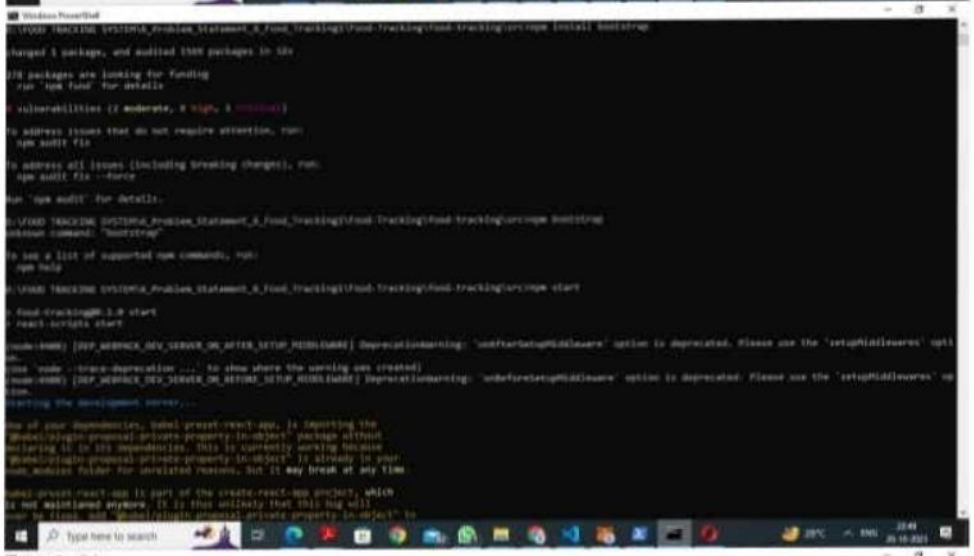


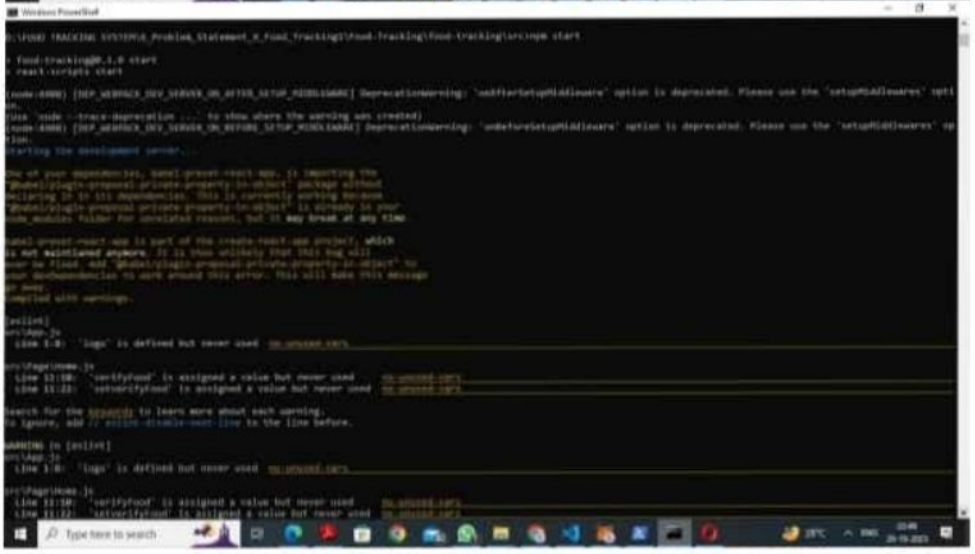

Gas estimation failed
Send Transaction
Cancel Transaction
Confirm transaction
Confirm
Cancel
DEPLOY & RUN TRANSACTIONS
Type here to search

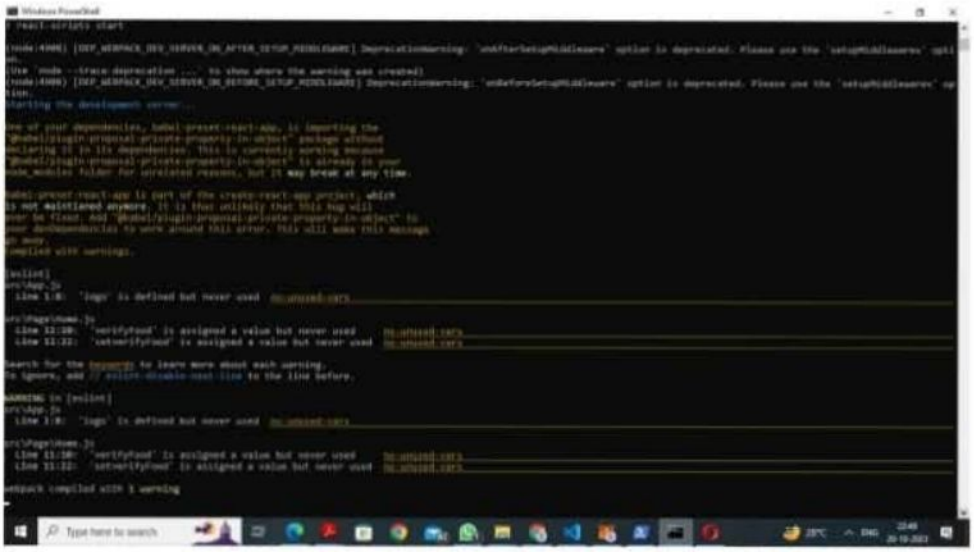
4.OPEN THE FILE EXPLORER:



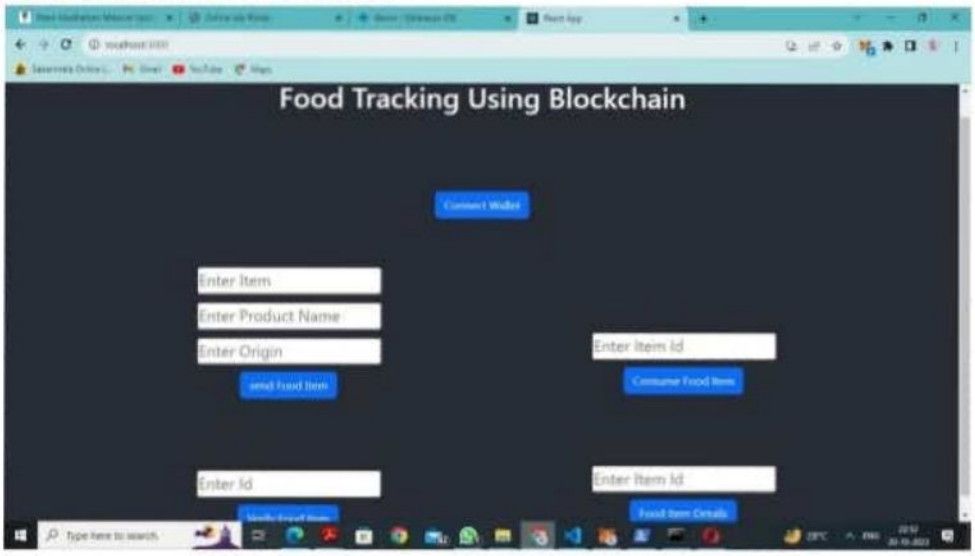




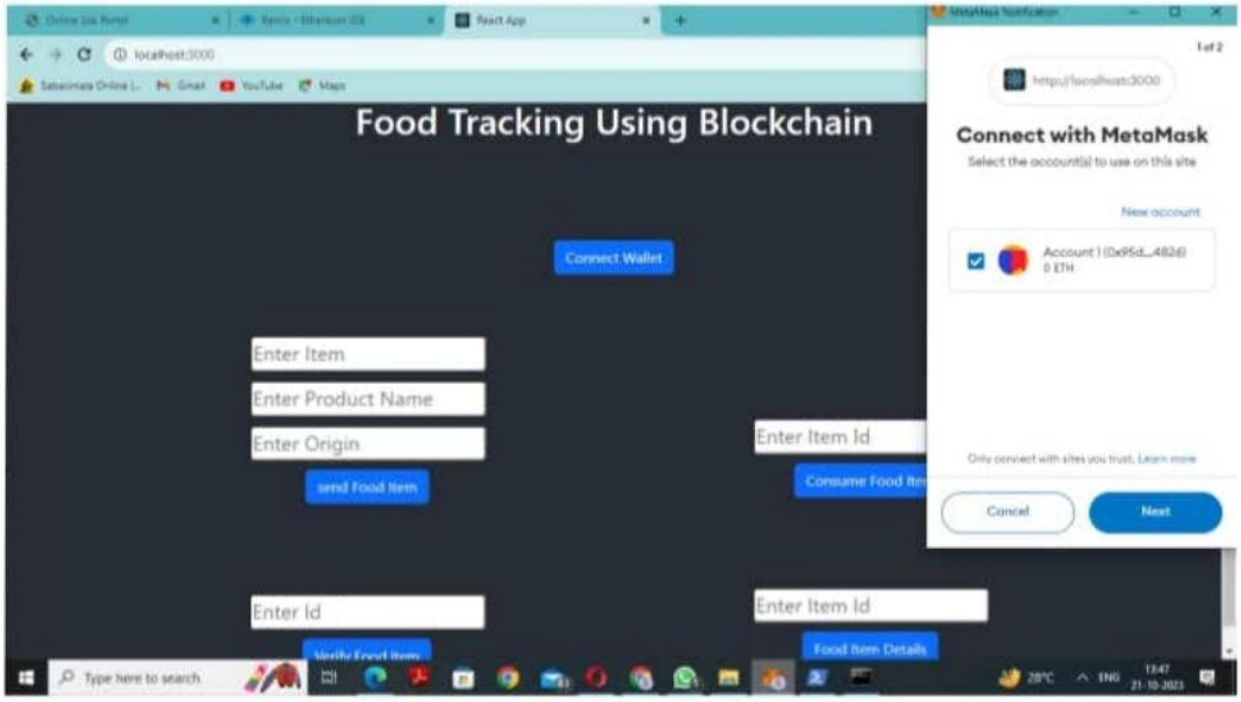


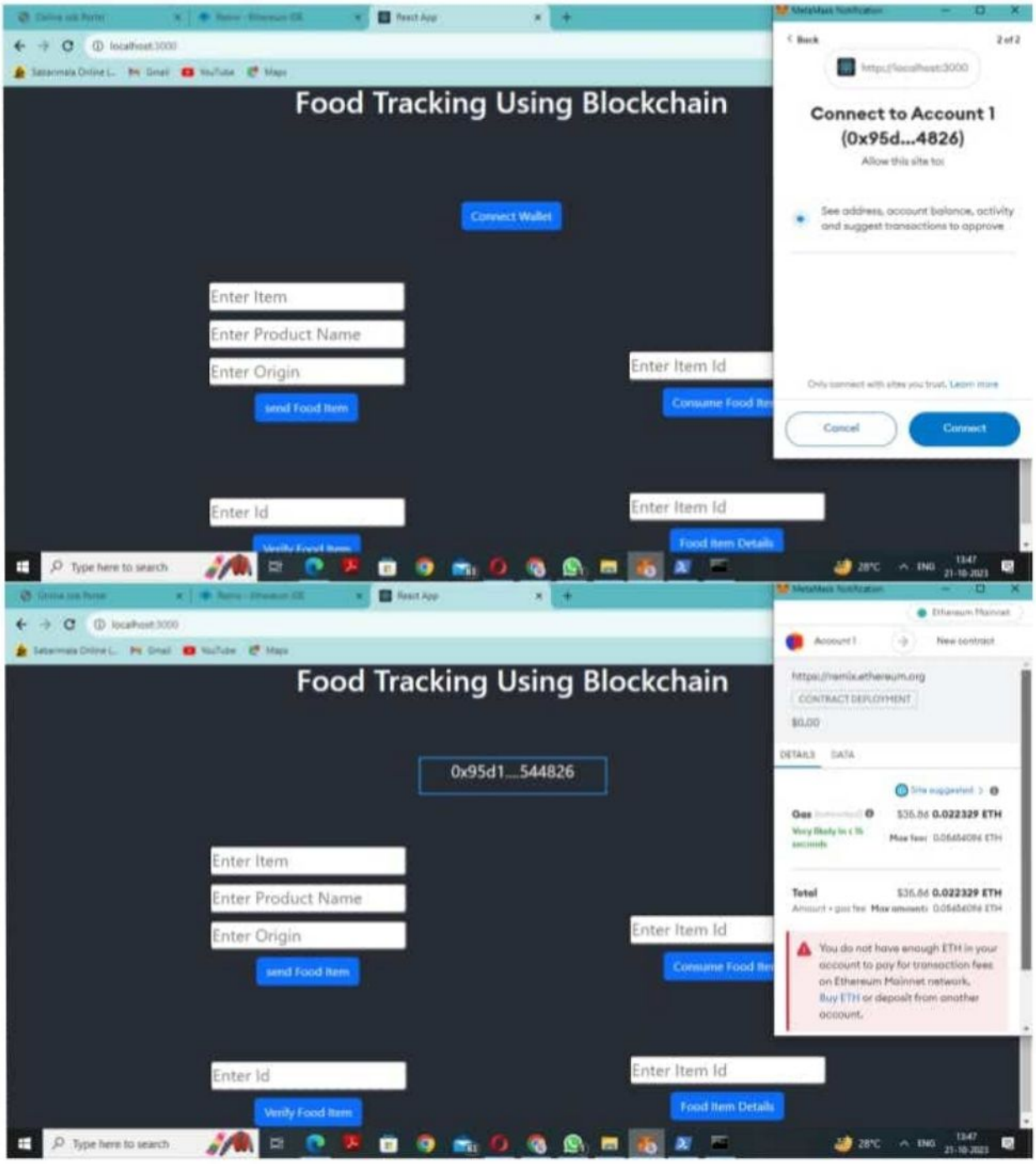


5.LOCALHOST IP ADDRESS:



RESULTS


React App
localhost:3000
Food Tracking Using Blockchain
Connect Wallet
Enter Item
Enter Product Name
Enter Origin
send Food Item
Enter Item Id
Consume Food Item
Enter Id
Verify Food Item
Enter Item Id
Food Item Details
Connect with MetaMask
Select the account(s) to use on this site
New account
Account 1 (0x95d...4824)
0 ETH
Only connect with sites you trust. Learn more
Cancel
Next
Type here to search


Food Tracking Using Blockchain
Connect Wallet
Enter Item
Enter Product Name
Enter Origin
send Food Item
Enter Item Id
Consume Food Item
Enter Id
Verify Food Item
Enter Item Id
Food Item Details
Connect to Account 1
(0x95d...4826)
Allow this site to:
See address, account balance, activity and suggest transactions to approve
Only connect with sites you trust. Learn more
Cancel
Connect
Food Tracking Using Blockchain
0x95d1....544826
Enter Item
Enter Product Name
Enter Origin
send Food Item
Enter Item Id
Consume Food Item
Enter Id
Verify Food Item
Enter Item Id
Food Item Details
https://remix.ethereum.org
CONTRACT DEPLOYMENT
$0.00
DETAILS
DATA
Gas
$36.84 0.022329 ETH
Total
$36.84 0.022329 ETH
You do not have enough ETH in your account to pay for transaction fees on Ethereum Mainnet network. Buy ETH or deposit from another account.

ADVANTAGES:

Benefits of using a blockchain food traceability system:

Accurate and tamper-resistant food data

Efficiently prevent, contain, or rectify contamination outbreaks while reducing the loss of revenue

Creates more transparency and trust in the authenticity of Verifiable Credential data

Secure and efficient data transfer between parties

Verifiable Credentials help eliminate and prevent fraud

Helps organizations comply with food regulations

Customers can instantly see information about food origins and product details

DISADVANTAGES:

High cost: Developing and implementing a blockchain-based supply chain management system can be expensive.

Complexity: Blockchain technology is still relatively new and can be complex to implement.

Lack of standardization: There is currently a lack of standardization in the blockchain industry, which can make it difficult for organizations to integrate blockchain technology

CONCLUSION:

Current food supply chain traceability systems are dominantly manual, expensive, have outdated software, are vulnerable to data manipulation, and lack data transparency for supply chain stakeholders. These problems often result in foodborne illness outbreaks that have caused many people to be sick or die. Fortunately blockchain, Verifiable Credential, and decentralized identifier (DID) technology can be integrated into existing systems and effectively solve many of these problems.

FUTURE SCOPE:

We see at least three food ecosystem issues where blockchain technology can make a vast contribution: ascertaining the origin of products, tracking ingredients, and improving sustainability (reporting). In all these cases, blockchain increases transparency or traceability.

The future of blockchain in finance is quite promising. The cost of money transfers between different intermediaries is very high. Blockchain technology can eliminate the need for such intermediaries and help in lowering the cost significantly. It can provide the finance sector with a transparent ledger system

APPENDIX:

REMIX IDE 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);
    }
}
```

SOURCE CODE:

```
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 = "0x95d1D247465fC4d91585f6e8d1870ae1B8544826"

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

GITHUB LINK

https://github.com/Logesh1625/food-tracking-system.git

DEMO LINK

https://drive.google.com/file/d/1QR3xwoaUJi7OkZKrL04E1cAtCRdozCMU/view?usp=drivesdk