A TRUSTED BLOCKCHAIN-BASED TRACEABILITY SYSTEM FOR FRUIT AND VEGETABLE AGRICULTURAL PRODUCTS

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE AWARD OF THE
DEGREE OF **BACHELOR OF TECHNOLOGY**
IN **INFORMATION TECHNOLOGY**
OF THE ANNA UNIVERSITY

PROJECT WORK

2023

Submitted by

DHARSHINI B
1918109
INDHIRANI S
1918112
JAYAMEENAKSHI S
1918115
JAYAPRIYA G
1918116
VARSHINI S
1918143

Under the Guidance of
Prof .S .Gladson Oliver M.Tech.,

DEPARTMENT OF INFORMATION TECHNOLOGY
GOVERNMENT COLLEGE OF TECHNOLOGY
(An Autonomous Institution affiliated to Anna University)
COIMBATORE - 641 013

DEPARTMENT OF INFORMATION TECHNOLOGY

GOVERNMENT COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University)

COIMBATORE - 641 013

PROJECT WORK

APRIL 2023

This is to certify that this project work entitled

A TRUSTED BLOCKCHAIN-BASED TRACEABILITY SYSTEM FOR FRUIT AND VEGETABLE AGRICULTURAL PRODUCTS

is the bonafide record of project work done by

DHARSHINI B

1918109

INDHIRANI S

1918112

JAYAMEENAKSHI S

1918115

JAYAPRIYA G

1918116

VARSHINI S

1918143

of **B.Tech**.(**Information Technology**) during the year 2022 – 2023

Prof. S. Gladson Oliver M.Tech.,	**Dr. S. Rathi M.E, Ph.D.,**
Project Guide	Head of the Department

Submitted for the Project Viva-Voce examination held on ____________

Internal Examiner	External Examiner

ACKNOWLEDGEMENT

Great achievements are not possible without standing on the shoulders ofgiants. Without the active involvement of the following experts this project would not have been a reality.

We express our sincere gratitude to **Dr. K.Manonmani M.E,Ph.D** Principal,Government College Of Technology, Coimbatore for providing us all facilities that we needed for the completion of this project.

We whole-heartedly express our thankfulness and gratitude to **Dr.S.Rathi M.E,Ph.D**., Associate professor and head of the Department of Information Technology, Government College Of Technology , for helping us to successfully carry out this project.

Our thankfulness and gratitude to our respectable project guide **Prof.S.Gladson Oliver M.Tech,** Assistant Professor, who has been helping through the various phases of the project. With his potent ideas and excellent guidance, we were able to comprehend the essential aspects involved.

We would like to thank our faculty advisor **Dr.R.Devi M.Tech,Ph.D.,** Assistant Professor for her continuous support and encouragement throughout this project.

We extend our sincere thanks to the staff members of the Information Technology department **Dr.R.Devi M.Tech,Ph.D.,** Assistant Professor, **Prof.C.Aswini M.Tech**., Assistant Professor, **Dr.T.Suguna** Assistant Professor, **Dr.M.BlessyQueen Mary M.E.,Ph.D**., Assistant Professor, **Prof.S.Gladson Oliver M.Tech**., Assistant Professor, **Prof.M.Jeyanthi M.Tech.**, Assistant Professor, **Dr.R.Malavika M.Tech**., Assistant Professor **Prof.M.Gowri Shankar M.E.,** Assistant Professor, for rendering their help for the completion of this project. We also thank all our friends for their cooperation and suggestions towards the successful completion of this project.

SYNOPSIS

In recent years, food safety issues have drawn growing concerns from society. In order to efficiently trace the accountability, building a reliable traceability system is indispensable. It is especially essential to accurately record, share and trace the specific data within the whole food supply chain including the process of production, processing, warehousing, transportation and retail. Traditional traceability systems have issues such as data invisibility, tampering and sensitive information disclosure. Blockchain is a promising technology for food safety traceability system because of the characteristics such as irreversible time vector, smart contract, etc. Furthermore, the smart contract is designed to prevent data tampering and sensitive information disclosure during information interaction among participants. The prototype system was implemented based on the Ethereum.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 DESCRIPTION

Traditional traceability system has problems of centralized management, opaque information, untrustworthy data, and easy generation of information islands. To solve the above problems, we design a traceability system based on blockchain technology for storage and query of product information in supply chain of agricultural products.

1.2 EXISTING SYSTEM

Traditional traceability technology is mainly through two-dimensional code or barcode technology, combined with radio frequency identification (RFID) technology. Through manual recording and storage in a centralized database, consumers can trace only the basic information of the product by scanning the barcode on the product package.

However, the process of grain food from farmland to dining table mainly includes the following links: production, distribution, warehousing and customer. Data in traditional traceability systems is centralized, and authoritative agencies manage the central database of the traceability system. Since the traceability data of each supply chain node are managed by enterprise, the data are easy to tamper with. Therefore, the reliability of information transmission among different roles in the agricultural supply chain needs to be increased.

1.3 PROBLEM DEFINITION

In recent years, food safety issues have drawn growing concerns from society. In order to efficiently trace the accountability, building a reliable traceability system is indispensable. It is especially essential to accurately record, share and trace the specific data within the whole food supply chain including the process of production, distribution, warehousing and until it finally reaches the customer.

1.4 PROPOSED SYSTEM

Blockchain as an emerging technology that has properties of decentralization, tamper-proof and traceability provides the possibility to solve the problems existing in the current traditional agricultural product traceability system.

A traceability system based on blockchain technology for storage and query of product information in supply chain of agricultural products. Leveraging the characteristics of decentralization, tamper-proof and traceability of blockchain technology, the transparency and credibility of traceability information increased.

1.5 ORGANISATION OF PROJECT

- Literature reviews of already existing proposals are discussed in chapter 2.
- Chapter 3 has system specification which tells about the software requirements.
- Chapter 4 discusses the overall project and design which tells the brief description of each of the modules in this project.
- Chapter 5 has the implementation and experimental result of the project.
- Chapter 6 deals with the conclusion.
- Finally, chapter 7 deals with the references.

CHAPTER 2

LITERATURE REVIEW

2.1 Blockchain-Based Soybean Traceability In Agricultural Supply Chain [1]

2.1.1 DESCRIPTION

It proposes on the utilization of smart contracts to govern and control all interactions and transactions among all the participants involved within the supply chain ecosystem.

2.1.2 MERITS

- All transactions are recorded and stored in the blockchain's immutable ledger with links to a decentralized file system (IPFS) and thus providing to all a high level of transparency and traceability into the supply chain ecosystem in a secure, trusted, reliable, and efficient manner.

2.1.3 DEMERITS

- There are key challenges related to scalability, governance, identity registration, privacy, standards, and regulations.

2.2 Smart Contract-Based Product Traceability System in the Supply Chain Scenario [2]

2.2.1 DESCRIPTION

A product traceability system based on blockchain technology, in which all product transferring histories are perpetually recorded in a distributed ledger by using smart contracts and a chain is formed that can trace back to the source of the products.

2.2.2 MERITS

- It has obvious decentralized characteristics, which significantly reduces the possibility of privately tampering with data within enterprises.

2.2.3 DEMERITS

- Possibility of manual input errors.

2.3 Blockchain-Based Traceability for the Fishery Supply Chain [3]

2.3.1 DESCRIPTION

A private Ethereum blockchain-based solution to efficiently manage the fishery supply chain operations in a manner that is decentralized, transparent, traceable, secure, private, and trustworthy. The solution architecture proposes five smart contracts.

2.3.2 MERITS

- It prevents different types of fish fraud and malpractice by ensuring transparent interactions among all stakeholders.

2.3.3 DEMERITS

- It does not build DApps for various stakeholders.

2.4 Blockchain-Based Safety Management System for the Grain Supply Chain [4]

2.4.1 DESCRIPTION

The grain supply chain is characterized by a long life cycle, complex links, various hazards, and heterogeneous information sources. Effective information management of the entire grain supply chain can improve information disclosure and sharing, reduce hazards in the production process, and ensure food safety.

2.4.2 MERITS

- The proposed information management system gives full play to blockchain’s advantages, avoids the problem of relying on core enterprises to collect information, makes information interaction among all links more open and transparent.

2.4.3 DEMERITS

- Blockchain information cannot be tampered with, stakeholders' impact on information authenticity is eliminated, and the phenomenon of artificial tampering is prevented.

2.5 Blockchain-Based Traceability System in Agri-Food SME: Case Study of a Traditional Bakery [5]

2.5.1 DESCRIPTION

The goal of the system is to guarantee a transparent and auditable traceability of the Carasau bread.

2.5.2 MERITS

- It could provide over existing solutions and for multiple fields and applications**.**

2.5.2 DEMERITS

- No complete functioning system that operates within the small regional industry.

2.6 Blockchain-Based Traceability System for Product Recall [6]

2.6.1 DESCRIPTION

To develop a traceability system integrated into the product recall system deployed to the Ethereum to ensure the transparency and visibility of the recall process for all stakeholders.

2.6.2 MERITS

- It achieves lower costs and increased economic worth.

2.6.3 DEMERITS

- It has low power efficiency and lacks user-friendly experience.

2.7 Blockchain-Based Agri-Food Supply Chain: A Complete Solution [7]

2.7.1 DESCRIPTION

The blockchain and smart contracts, deployed over ethereum blockchain network. All transactions are written to blockchain which ultimately uploads the data to IPFS.

2.7.2 MERITS

- The simulations and evaluation of smart contracts along with the security and vulnerability are analysed.

2.7.3 DEMERITS

- It fails to solve some major problems in supply chain management like credibility of the involved entities, accountability of the trading process and traceability of the products.

CHAPTER 3
SYSTEM SPECIFICATION

3.1 SYSTEM REQUIREMENTS

3.1.1 SOFTWARE REQUIREMENTS

Operating System	:	Windows 11
Coding Language	:	Solidity
Tool	:	Visual Studio Code, Ganache UI
Frontend	:	ReactJS, Material UI
Backend	:	NodeJS, JavaScript, Solidity

3.2 SOFTWARE DESCRIPTION

3.2.1 ABOUT VISUAL STUDIO CODE

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, Solidity, PHP, Go, .NET).

3.2.2 TRUFFLE

Truffle is a popular development framework for Ethereum that provides a suite of tools and libraries to help developers build, test, and deploy smart contracts and decentralized applications (dapps) on the Ethereum blockchain. It provides several built-in features, such as contract compilation, testing, deployment, and migration. It also provides contract management, automated deployment scripts.

3.2.3 GANACHE

Ganache is a personal blockchain for Ethereum development. It allows developers to test their smart contracts and applications in a local and isolated environment before deploying them to the main Ethereum network. It allows for rapid development and testing.

Ganache also provides a user-friendly interface that allows developers to interact with their local blockchain using a graphical user interface (GUI). This interface allows you to see the status of your blockchain, view transaction details, and monitor network activity.

3.2.4 SOLIDITY

Solidity is a programming language used to write smart contracts on the Ethereum blockchain platform. It is a high-level language that is similar to JavaScript and is designed to be easy to learn and use.

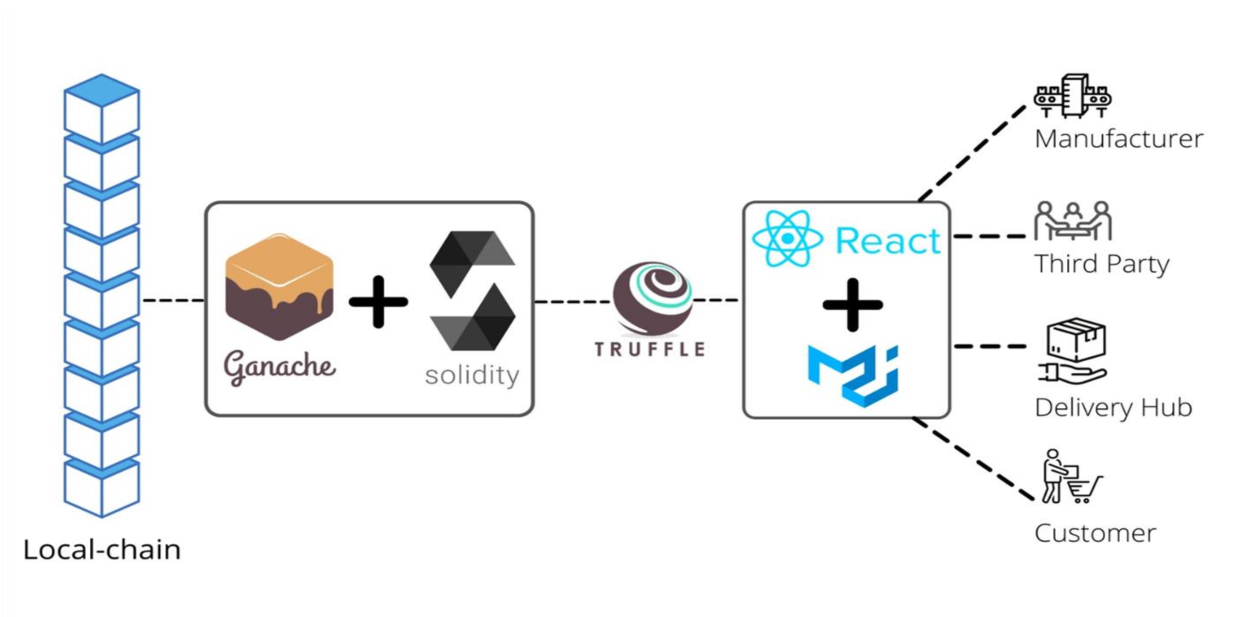
Smart contracts are self-executing programs that run on the Ethereum blockchain and can be programmed to perform a wide range of tasks, such as managing digital assets, executing financial transactions, and implementing complex business logic.

3.2.5 CHARACTERISTICS OF SOLIDITY

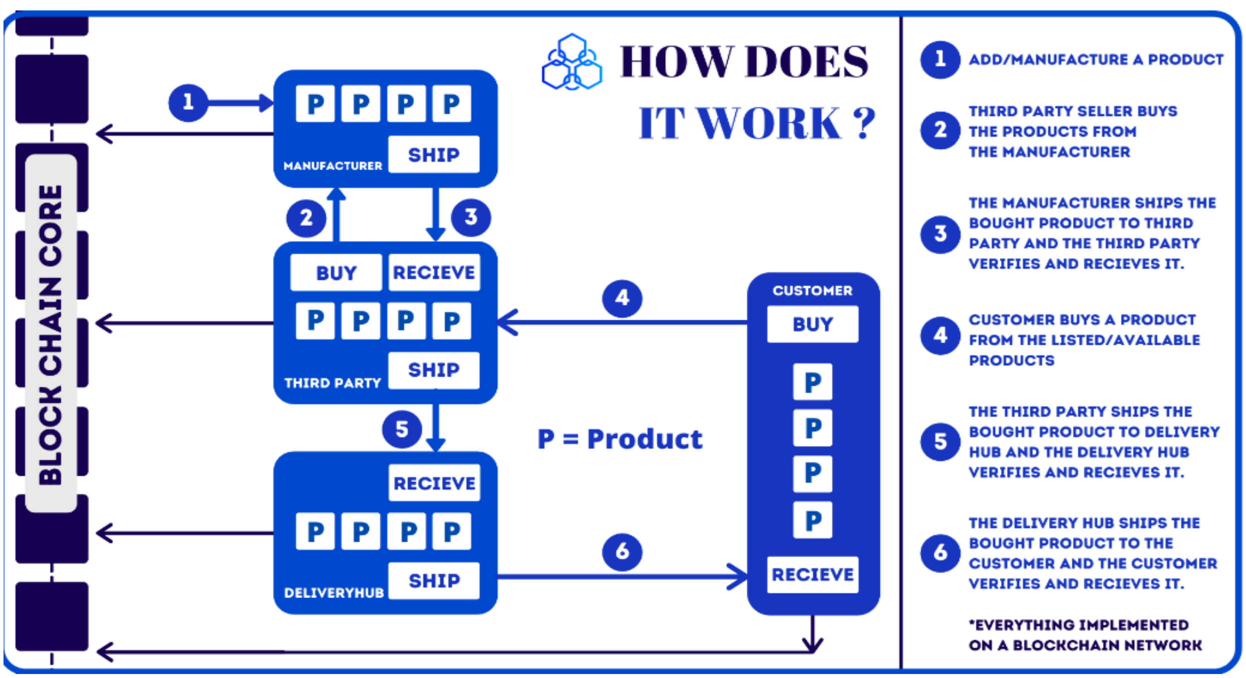
- Object-oriented
- Strongly typed
- Contract-based
- Gas-based
- Contract-oriented

CHAPTER 4
PROJECT DESIGN

4.1 ARCHITECTURE DIAGRAM



4.2 FLOW CHART



4.3 MODULE DESCRIPTION

4.3.1 MANUFACTURER MODULE

The lifecycle of a product starts when manufactureProduct() is called(while making an entry) after the final product is manufactured and the product and manufacturer details are entered in the blockchain. The productHistory[] gets initialized and the current product data is stored with the current owner(manufacturer).

4.3.2 THIRD-PARTY MODULE

Now this product shall be available to the Third Party for purchase. On being purchased by a third party seller, the purchasedByThirdParty() gets called where the owner is set to thirdParty and the present data gets pushed to the productHistory[] (which helps us to track the origin and handling of the product). Simultaneously, the product is shipped by the manufacturer (shipToThirdParty()) and is received by the Third Party where receivedByThirdParty() is called and the details of the Third Party seller are entered. Each of these checkpoint's data is stored in product history with the state being updated at each step.

4.3.3 DELIVERY HUB MODULE

The online purchase of the product takes place from the Third Party. When the customer orders the product, it is shipped by the Third Party (shipByThirdParty()) and received by the delivery hub where the receivedByDeliveryHub() is called. Here the customer address is stored, owner is set to Delivery Hub, details of the Delivery Hub are fed and the current data state gets pushed to the productHistory[].

4.3.4 CUSTOMER MODULE

Finally the product is shipped by the Delivery Hub (shipByDeliveryHub()) and received by the customer where the receivedByCustomer() is called and the current and final state gets pushed to the productHistory[].

All of these juncture functions shall be called only after complete verification of product and productHistory[] while entering a checkpoint. (eg:- Customer accepts and confirms the product by clicking the receive button from his account only after it verifies the product).

fetchProductPart1(), fetchProductPart2(), fetchProductPart3(), fetchProductHistoryLength(), fetchProductCount(), fetchProductState() are the functions to retreive data of a product queried with UID and data type as product(current state) or history.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

5.1.1 FRONTEND REACT COMPONENT

MAP.JS

```
import React, { Component } from 'react';
import { Map, GoogleApiWrapper, Marker } from 'google-maps-react';
const mapStyles = {
  width: '95%',
  borderRadius:"10px",
  height: "350px",
  zIndex: "10 !important",
  border:"2px solid #1a237e",
};

export class MapContainer extends Component {
  constructor(props) {
    super(props);
    var points =[];
    if(props.prodData[0][7].length !== 0) { points.push({latitude: props.prodData[0][7],
longitude: props.prodData[0][6]})};
    if(props.prodData[2][0].length !== 0) {points.push( {latitude: props.prodData[2][0],
longitude: props.prodData[1][7]})};
    if(props.prodData[2][3].length !== 0) {points.push({latitude: props.prodData[2][3],
longitude: props.prodData[2][2]})};

    this.state = {
      stores: points
    }
  }

  displayMarkers = () => {
    return this.state.stores.map((store, index) => {
      return <Marker key={index} id={index} position={{
       lat: store.latitude,
       lng: store.longitude
     }}

     onClick={() => console.log("You clicked me!")} />
    })
  }
```

```
  render() {
    return (
        <Map
          google={this.props.google}
          zoom={8}
          style={mapStyles}
          initialCenter={{ lat: 11.018666368489749, lng: 76.93596653946759}}
        >
          {this.displayMarkers()}
        </Map>
    );
  }
}

MapContainer = GoogleApiWrapper({
  apiKey: process.env.REACT_APP_GOOGLE_MAP_API_KEY
})(MapContainer);
```

5.1.2 BACKEND SMART CONTRACTS

5.1.2.1 STRUCTURE.SOL

```
pragma solidity >=0.4.21 <0.9.0;

library Structure {
    enum State {
        Manufactured,
        PurchasedByThirdParty,
        ShippedByManufacturer,
        ReceivedByThirdParty,
        PurchasedByCustomer,
        ShippedByThirdParty,
        ReceivedByDeliveryHub,
        ShippedByDeliveryHub,
        ReceivedByCustomer
    }
    struct ManufactureDetails {
        address manufacturer;
        string manufacturerName;
        string manufacturerDetails;
        string manufacturerLongitude;
        string manufacturerLatitude;
        uint256 manufacturedDate;
    }
```

```
    struct ProductDetails {
        string productName;
        uint256 productCode;
        uint256 productPrice;
        string productCategory;
    }
    struct ThirdPartyDetails {
        address thirdParty;
        string thirdPartyLongitude;
        string thirdPartyLatitude;
    }
    struct DeliveryHubDetails {
        address deliveryHub;
        string deliveryHubLongitude;
        string deliveryHubLatitude;
    }
    struct Product {
        uint256 uid;
        uint256 sku;
        address owner;
        State productState;
        ManufactureDetails manufacturer;
        ThirdPartyDetails thirdparty;
        ProductDetails productdet;
        DeliveryHubDetails deliveryhub;
        address customer;
        string transaction;
    }

    struct ProductHistory {
        Product[] history;
    }

    struct Roles {
        bool Manufacturer;
        bool ThirdParty;
        bool DeliveryHub;
        bool Customer;
    }
}
```

5.1.2.2 ROLES.SOL

```
pragma solidity >=0.4.21 <0.9.0;

library Roles{
```

```
  struct Role {
    mapping (address => bool) list;
  }

  function hasRole(Role storage role, address _account)
    internal
    view
    returns (bool)
  {
    require(_account != address(0));
    return role.list[_account];
  }

  function addRole(Role storage role, address _account)
    internal
  {
    require(_account != address(0));
    require(!hasRole(role, _account));

    role.list[_account] = true;
  }

}
```

5.1.2.3 MANUFACTURER.SOL

```
pragma solidity >=0.4.21 <0.9.0;

import "./Roles.sol";

contract Manufacturer {
  using Roles for Roles.Role;

  event ManufacturerAdded(address indexed _account);

  Roles.Role  manufacturersList;

  constructor() public {
    manufacturersList.addRole(msg.sender);
    emit ManufacturerAdded(msg.sender);
  }

  modifier onlyManufacturer() {
    require(isManufacturer(msg.sender));
    _;
  }
```

```
  function isManufacturer(address _account) public view returns (bool) {
    return manufacturersList.hasRole(_account);
  }

  function addManufacturer(address _account ) public {
    manufacturersList.addRole(_account);
    emit ManufacturerAdded(_account);
  }

}
```

5.1.2.4 THIRDPARTY.SOL

```
pragma solidity >=0.4.21 <0.9.0;

import "./Roles.sol";

contract Thirdparty {
  using Roles for Roles.Role;

  event ThirdpartyAdded(address indexed _account);

  Roles.Role private thirdpartysList;

  constructor() public {
    thirdpartysList.addRole(msg.sender);
    emit ThirdpartyAdded(msg.sender);
  }

  modifier onlyThirdparty() {
    require(isThirdparty(msg.sender));
    _;
  }

  function isThirdparty(address _account) public view returns (bool) {
    return thirdpartysList.hasRole(_account);
  }

  function addThirdparty(address _account) public onlyThirdparty {
    thirdpartysList.addRole(_account);
    emit ThirdpartyAdded(_account);
  }
}
```

5.1.2.5 DELIVERYHUB.SOL

```
pragma solidity >=0.4.21 <0.9.0;

import "./Roles.sol";

contract DeliveryHub {
  using Roles for Roles.Role;

  event DeliveryHubAdded(address indexed _account);

  Roles.Role private deliveryHubsList;

  constructor() public {
    deliveryHubsList.addRole(msg.sender);
    emit DeliveryHubAdded(msg.sender);
  }

  modifier onlyDeliveryHub() {
    require(isDeliveryHub(msg.sender));
    _;
  }

  function isDeliveryHub(address _account) public view returns (bool) {
    return deliveryHubsList.hasRole(_account);
  }

  function addDeliveryHub(address _account) public onlyDeliveryHub {
    deliveryHubsList.addRole(_account);
    emit DeliveryHubAdded(_account);
  }
}
```

5.1.2.6 CUSTOMER.SOL

```
pragma solidity >=0.4.21 <0.9.0;

import "./Roles.sol";

contract Customer {
  using Roles for Roles.Role;

  event CustomerAdded(address indexed _account);
  Roles.Role private customersList;
```

```
  constructor() public {
    customersList.addRole(msg.sender);
    emit CustomerAdded(msg.sender);
  }

  modifier onlyCustomer() {
    require(isCustomer(msg.sender));
    _;
  }

  function isCustomer(address _account) public view returns (bool) {
    return customersList.hasRole(_account);
  }

  function addCustomer(address _account) public onlyCustomer {
    customersList.addRole(_account);
    emit CustomerAdded(_account);
  }
}
```

5.1.2.7 SUPPLYCHAIN.SOL

```
pragma solidity >=0.4.21 <0.9.0;

import "./Structure.sol";

contract SupplyChain {
    event ManufacturerAdded(address indexed _account);

    //product code
    uint256 public uid;
    uint256 sku;

    address owner;

    mapping(uint256 => Structure.Product) products;
    mapping(uint256 => Structure.ProductHistory) productHistory;
    mapping(address => Structure.Roles) roles;

    function hasManufacturerRole(address _account) public view returns (bool) {
        require(_account != address(0));
        return roles[_account].Manufacturer;
    }

    function addManufacturerRole(address _account) public {
        require(_account != address(0));
```

```
    require(!hasManufacturerRole(_account));

    roles[_account].Manufacturer = true;
}

function hasThirdPartyRole(address _account) public view returns (bool) {
    require(_account != address(0));
    return roles[_account].ThirdParty;
}

function addThirdPartyRole(address _account) public {
    require(_account != address(0));
    require(!hasThirdPartyRole(_account));

    roles[_account].ThirdParty = true;
}

function hasDeliveryHubRole(address _account) public view returns (bool) {
    require(_account != address(0));
    return roles[_account].DeliveryHub;
}

function addDeliveryHubRole(address _account) public {
    require(_account != address(0));
    require(!hasDeliveryHubRole(_account));

    roles[_account].DeliveryHub = true;
}

function hasCustomerRole(address _account) public view returns (bool) {
    require(_account != address(0));
    return roles[_account].Customer;
}

function addCustomerRole(address _account) public {
    require(_account != address(0));
    require(!hasDeliveryHubRole(_account));

    roles[_account].Customer = true;
}

constructor() public payable {
    owner = msg.sender;
    sku = 1;
    uid = 1;
}
```

```
event Manufactured(uint256 uid);
event PurchasedByThirdParty(uint256 uid);
event ShippedByManufacturer(uint256 uid);
event ReceivedByThirdParty(uint256 uid);
event PurchasedByCustomer(uint256 uid);
event ShippedByThirdParty(uint256 uid);
event ReceivedByDeliveryHub(uint256 uid);
event ShippedByDeliveryHub(uint256 uid);
event ReceivedByCustomer(uint256 uid);

modifier verifyAddress(address add) {
   require(msg.sender == add);
   _;
}

modifier manufactured(uint256 _uid) {
   require(products[_uid].productState == Structure.State.Manufactured);
   _;
}

modifier shippedByManufacturer(uint256 _uid) {
   require(
      products[_uid].productState == Structure.State.ShippedByManufacturer
   );
   _;
}

modifier receivedByThirdParty(uint256 _uid) {
   require(
      products[_uid].productState == Structure.State.ReceivedByThirdParty
   );
   _;
}

modifier purchasedByCustomer(uint256 _uid) {
   require(
      products[_uid].productState == Structure.State.PurchasedByCustomer
   );
   _;
}

modifier shippedByThirdParty(uint256 _uid) {
   require(
      products[_uid].productState == Structure.State.ShippedByThirdParty
   );
```

```
    _;
}

modifier receivedByDeliveryHub(uint256 _uid) {
    require(
        products[_uid].productState == Structure.State.ReceivedByDeliveryHub
    );
    _;
}

modifier shippedByDeliveryHub(uint256 _uid) {
    require(
        products[_uid].productState == Structure.State.ShippedByDeliveryHub
    );
    _;
}

modifier receivedByCustomer(uint256 _uid) {
    require(
        products[_uid].productState == Structure.State.ReceivedByCustomer
    );
    _;
}

function manufactureEmptyInitialize(Structure.Product memory product)
    internal
    pure
{
    address thirdParty;
    string memory transaction;
    string memory thirdPartyLongitude;
    string memory thirdPartyLatitude;

    address deliveryHub;
    string memory deliveryHubLongitude;
    string memory deliveryHubLatitude;
    address customer;

    product.thirdparty.thirdParty = thirdParty;
    product.thirdparty.thirdPartyLongitude = thirdPartyLongitude;
    product.thirdparty.thirdPartyLatitude = thirdPartyLatitude;

    product.deliveryhub.deliveryHub = deliveryHub;
    product.deliveryhub.deliveryHubLongitude = deliveryHubLongitude;
    product.deliveryhub.deliveryHubLatitude = deliveryHubLatitude;
```

```
    product.customer = customer;
    product.transaction = transaction;
}

function manufactureProductInitialize(
    Structure.Product memory product,
    string memory productName,
    uint256 productCode,
    uint256 productPrice,
    string memory productCategory
) internal pure {
    product.productdet.productName = productName;
    product.productdet.productCode = productCode;
    product.productdet.productPrice = productPrice;
    product.productdet.productCategory = productCategory;
}


function manufactureProduct(
    string memory manufacturerName,
    string memory manufacturerDetails,
    string memory manufacturerLongitude,
    string memory manufacturerLatitude,
    string memory productName,
    uint256 productCode,
    uint256 productPrice,
    string memory productCategory
) public {
    require(hasManufacturerRole(msg.sender));
    uint256 _uid = uid;
    Structure.Product memory product;
    product.sku = sku;
    product.uid = _uid;
    product.manufacturer.manufacturerName = manufacturerName;
    product.manufacturer.manufacturerDetails = manufacturerDetails;
    product.manufacturer.manufacturerLongitude = manufacturerLongitude;
    product.manufacturer.manufacturerLatitude = manufacturerLatitude;
    product.manufacturer.manufacturedDate = block.timestamp;
    product.owner = msg.sender;
    product.manufacturer.manufacturer = msg.sender;
    manufactureEmptyInitialize(product);
    product.productState = Structure.State.Manufactured;
    manufactureProductInitialize(
        product,
        productName,
        productCode,
```

```
        productPrice,
        productCategory
    );
    products[_uid] = product;
    productHistory[_uid].history.push(product);
    sku++;
    uid = uid + 1;
    emit Manufactured(_uid);
}

function purchaseByThirdParty(uint256 _uid) public manufactured(_uid) {
    require(hasThirdPartyRole(msg.sender));
    products[_uid].thirdparty.thirdParty = msg.sender;
    products[_uid].productState = Structure.State.PurchasedByThirdParty;
    productHistory[_uid].history.push(products[_uid]);

    emit PurchasedByThirdParty(_uid);
}

function shipToThirdParty(uint256 _uid)
    public
    verifyAddress(products[_uid].manufacturer.manufacturer)
{
    require(hasManufacturerRole(msg.sender));
    products[_uid].productState = Structure.State.ShippedByManufacturer;
    productHistory[_uid].history.push(products[_uid]);

    emit ShippedByManufacturer(_uid);
}

function receiveByThirdParty(
    uint256 _uid,
    string memory thirdPartyLongitude,
    string memory thirdPartyLatitude
)
    public
    shippedByManufacturer(_uid)
    verifyAddress(products[_uid].thirdparty.thirdParty)
{
    require(hasThirdPartyRole(msg.sender));
    products[_uid].owner = msg.sender;
    products[_uid].thirdparty.thirdPartyLongitude = thirdPartyLongitude;
    products[_uid].thirdparty.thirdPartyLatitude = thirdPartyLatitude;
    products[_uid].productState = Structure.State.ReceivedByThirdParty;
    productHistory[_uid].history.push(products[_uid]);
```

```
        emit ReceivedByThirdParty(_uid);
    }

    function purchaseByCustomer(uint256 _uid)
        public
        receivedByThirdParty(_uid)
    {
        require(hasCustomerRole(msg.sender));
        products[_uid].customer = msg.sender;
        products[_uid].productState = Structure.State.PurchasedByCustomer;
        productHistory[_uid].history.push(products[_uid]);

        emit PurchasedByCustomer(_uid);
    }

    function shipByThirdParty(uint256 _uid)
        public
        verifyAddress(products[_uid].owner)
        verifyAddress(products[_uid].thirdparty.thirdParty)
    {
        require(hasThirdPartyRole(msg.sender));
        products[_uid].productState = Structure.State.ShippedByThirdParty;
        productHistory[_uid].history.push(products[_uid]);

        emit ShippedByThirdParty(_uid);
    }

    function receiveByDeliveryHub(
        uint256 _uid,
        string memory deliveryHubLongitude,
        string memory deliveryHubLatitude
    ) public shippedByThirdParty(_uid) {
        require(hasDeliveryHubRole(msg.sender));
        products[_uid].owner = msg.sender;
        products[_uid].deliveryhub.deliveryHub = msg.sender;
        products[_uid].deliveryhub.deliveryHubLongitude = deliveryHubLongitude;
        products[_uid].deliveryhub.deliveryHubLatitude = deliveryHubLatitude;
        products[_uid].productState = Structure.State.ReceivedByDeliveryHub;
        productHistory[_uid].history.push(products[_uid]);
        emit ReceivedByDeliveryHub(_uid);
  }
   function shipByDeliveryHub(uint256 _uid)
        public
        receivedByDeliveryHub(_uid)
        verifyAddress(products[_uid].owner)
        verifyAddress(products[_uid].deliveryhub.deliveryHub)
```

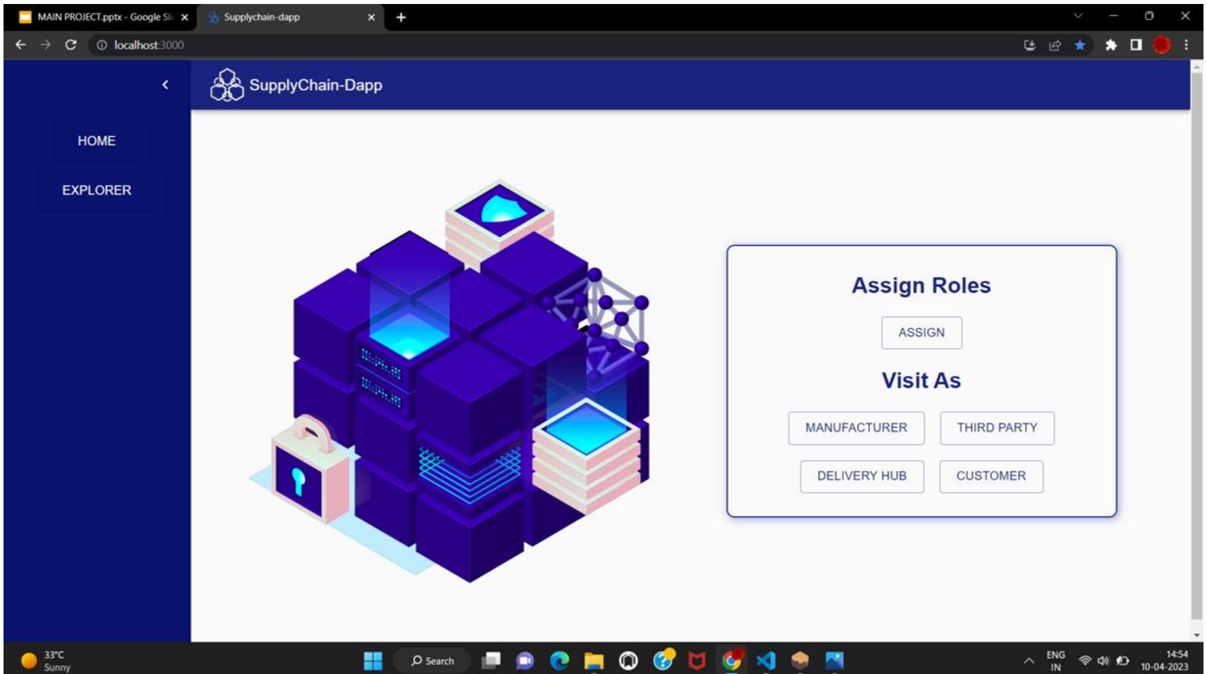
```
{
   require(hasDeliveryHubRole(msg.sender));
   products[_uid].productState = Structure.State.ShippedByDeliveryHub;
   productHistory[_uid].history.push(products[_uid]);

   emit ShippedByDeliveryHub(_uid);
}

function receiveByCustomer(uint256 _uid)
   public
   shippedByDeliveryHub(_uid)
   verifyAddress(products[_uid].customer)
{
   require(hasCustomerRole(msg.sender));
   products[_uid].owner = msg.sender;
   products[_uid].productState = Structure.State.ReceivedByCustomer;
   productHistory[_uid].history.push(products[_uid]);

   emit ReceivedByCustomer(_uid);
}
function fetchProductPart1(
   uint256 _uid,
   string memory _type,
   uint256 i
)
   public
   view
   returns (
      uint256,
      uint256,
      address,
      address,
      string memory,
      string memory,
      string memory,
      string memory
   )
{
   require(products[_uid].uid != 0);
   Structure.Product storage product = products[_uid];
   if (keccak256(bytes(_type)) == keccak256(bytes("product"))) {
      product = products[_uid];
   }
   if (keccak256(bytes(_type)) == keccak256(bytes("history"))) {
      product = productHistory[_uid].history[i];
   }
```

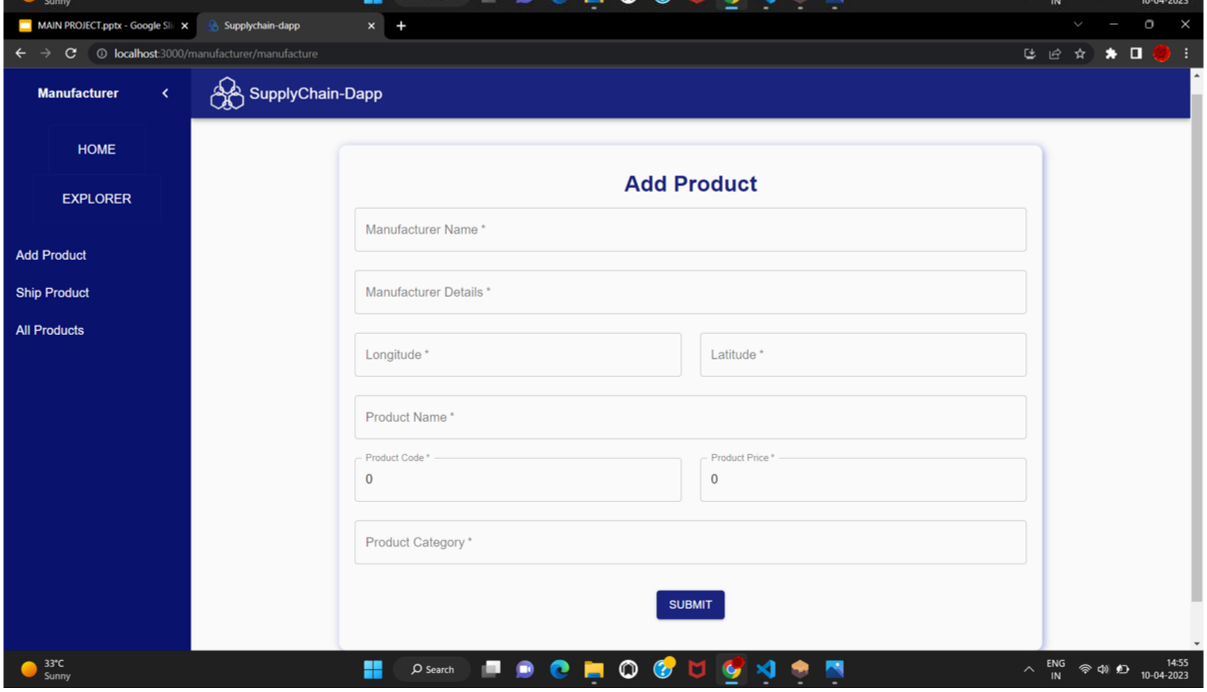
```
    return (
        product.uid,
        product.sku,
        product.owner,
        product.manufacturer.manufacturer,
        product.manufacturer.manufacturerName,
        product.manufacturer.manufacturerDetails,
        product.manufacturer.manufacturerLongitude,
        product.manufacturer.manufacturerLatitude
    );
}
function fetchProductPart2(
    uint256 _uid,
    string memory _type,
    uint256 i
)
    public
    view
    returns (
        uint256,
        string memory,
        uint256,
        uint256,
        string memory,
        Structure.State,
        address,
        string memory
    )
{
    require(products[_uid].uid != 0);
    Structure.Product storage product = products[_uid];
    if (keccak256(bytes(_type)) == keccak256(bytes("product"))) {
        product = products[_uid];
    }
    if (keccak256(bytes(_type)) == keccak256(bytes("history"))) {
        product = productHistory[_uid].history[i];
    }
    return (
        product.manufacturer.manufacturedDate,
        product.productdet.productName,
        product.productdet.productCode,
        product.productdet.productPrice,
        product.productdet.productCategory,
        product.productState,
        product.thirdparty.thirdParty,
        product.thirdparty.thirdPartyLongitude
```

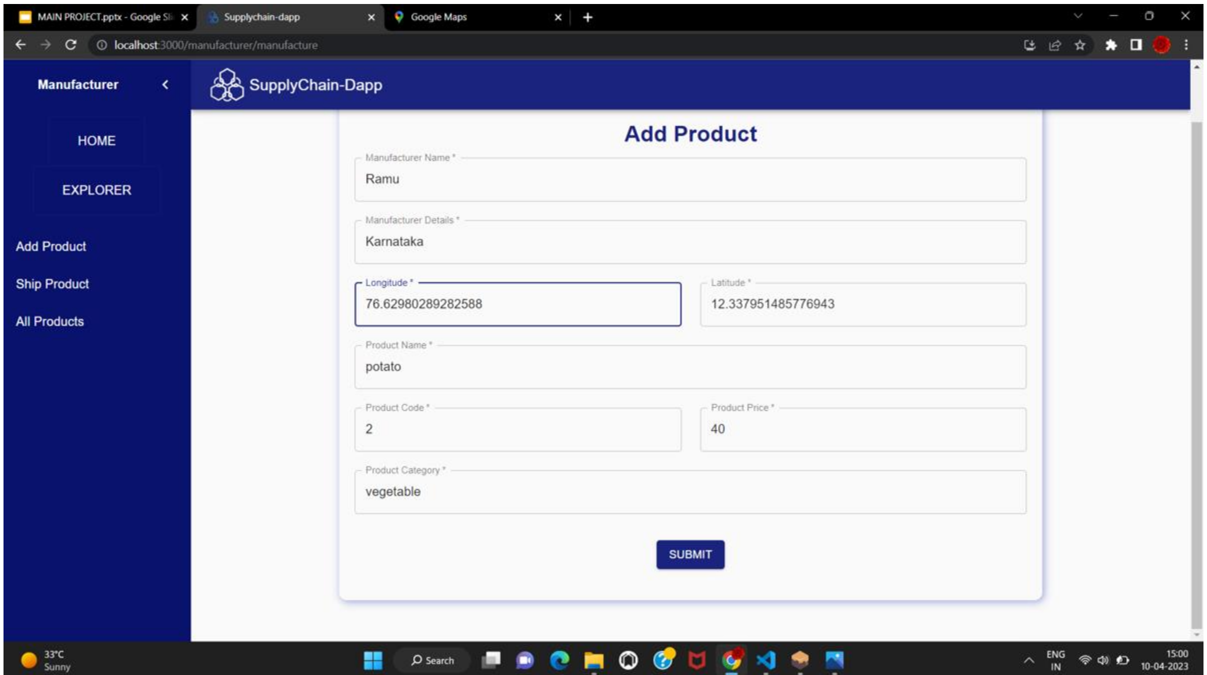
```
    );
}

function fetchProductPart3(
    uint256 _uid,
    string memory _type,
    uint256 i
)
    public
    view
    returns (
        string memory,
        address,
        string memory,
        string memory,
        address,
        string memory
    )
{
    require(products[_uid].uid != 0);
    Structure.Product storage product = products[_uid];
    if (keccak256(bytes(_type)) == keccak256(bytes("product"))) {
        product = products[_uid];
    }
    if (keccak256(bytes(_type)) == keccak256(bytes("history"))) {
        product = productHistory[_uid].history[i];
    }
    return (
        product.thirdparty.thirdPartyLatitude,
        product.deliveryhub.deliveryHub,
        product.deliveryhub.deliveryHubLongitude,
        product.deliveryhub.deliveryHubLatitude,
        product.customer,
        product.transaction
    );
}

function fetchProductCount() public view returns (uint256) {
    return uid;
}

function fetchProductHistoryLength(uint256 _uid)
    public
    view
    returns (uint256)
{
```

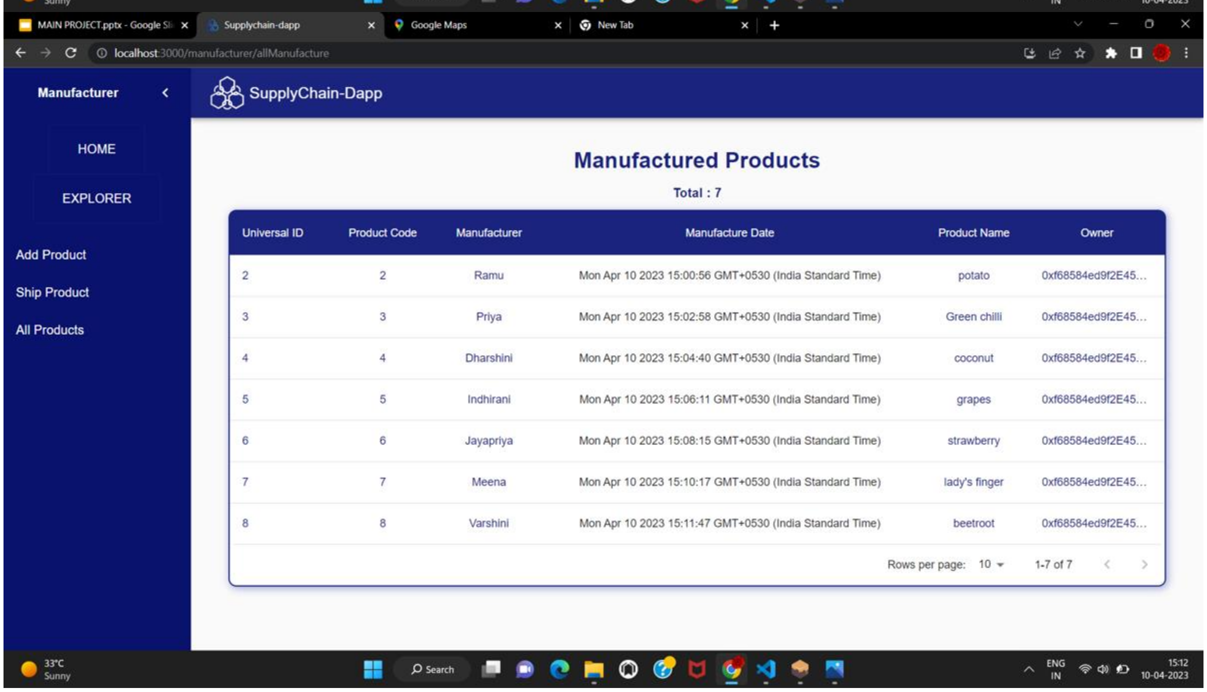
```
        return productHistory[_uid].history.length;
    }

    function fetchProductState(uint256 _uid)
        public
        view
        returns (Structure.State)
    {
        return products[_uid].productState;
    }

    function setTransactionHashOnManufacture(string memory tran) public {
        productHistory[uid - 1].history[
            productHistory[uid - 1].history.length - 1
        ]
            .transaction = tran;
    }

    function setTransactionHash(uint256 _uid, string memory tran) public {
        Structure.Product storage p =
            productHistory[_uid].history[
                productHistory[_uid].history.length - 1
            ];
        p.transaction = tran;
    }
}
```

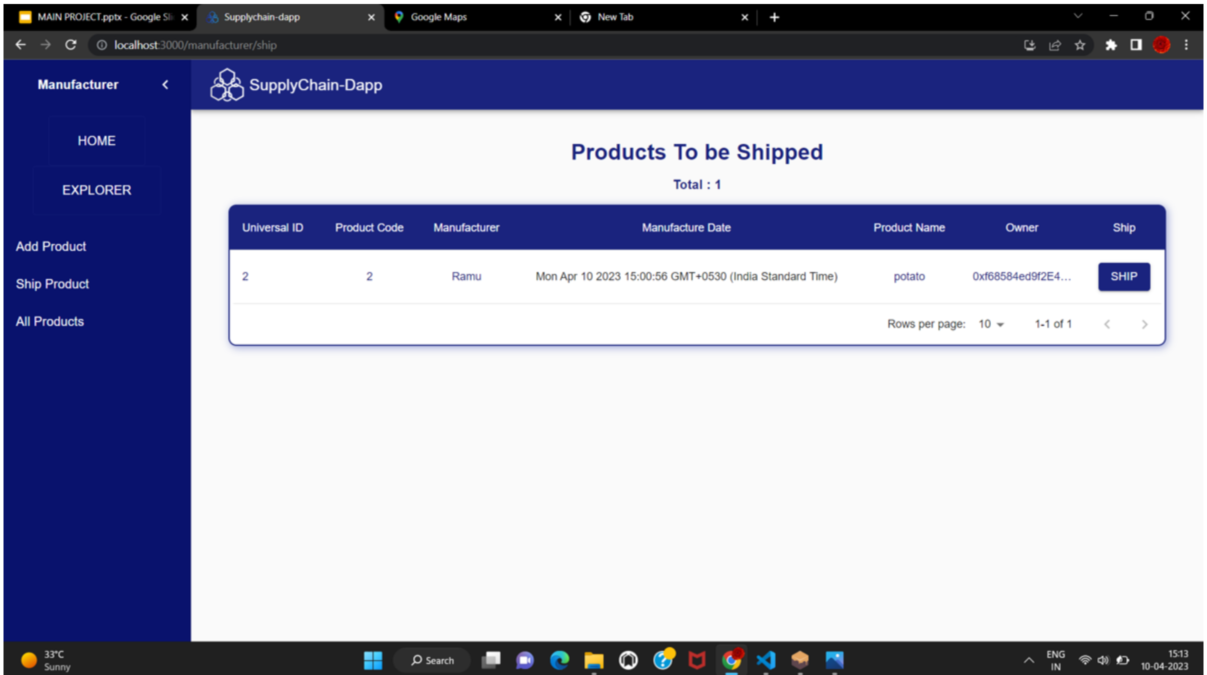
5.2 SAMPLE OUTPUT

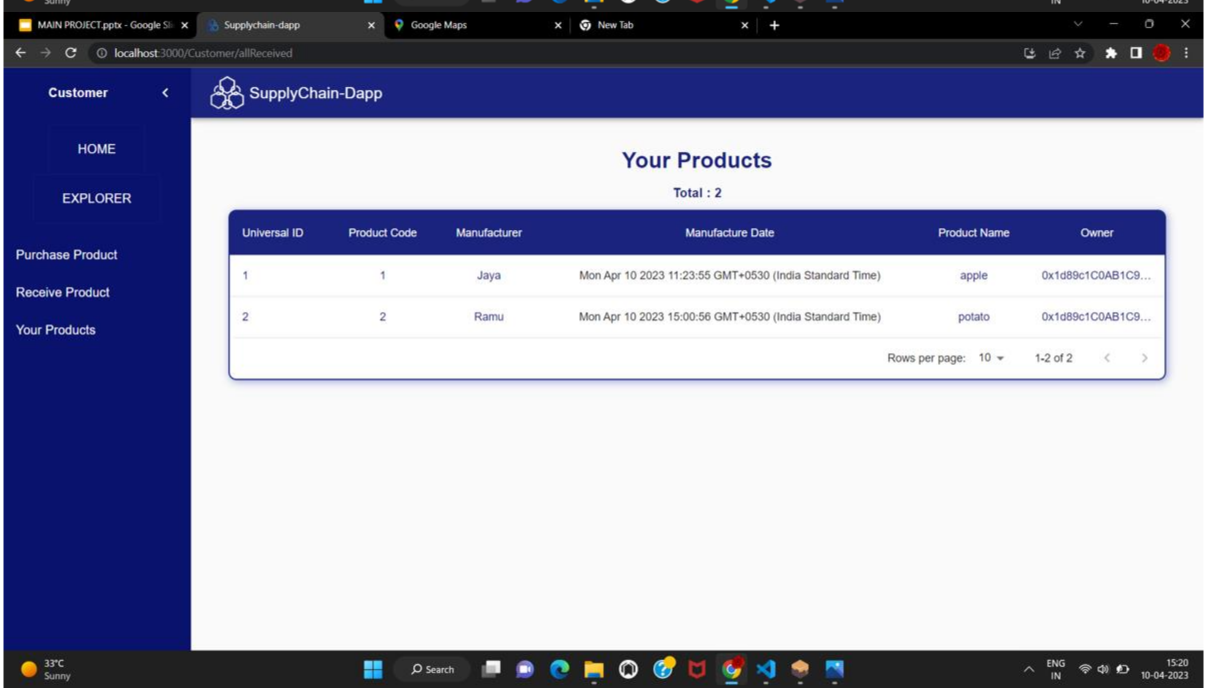


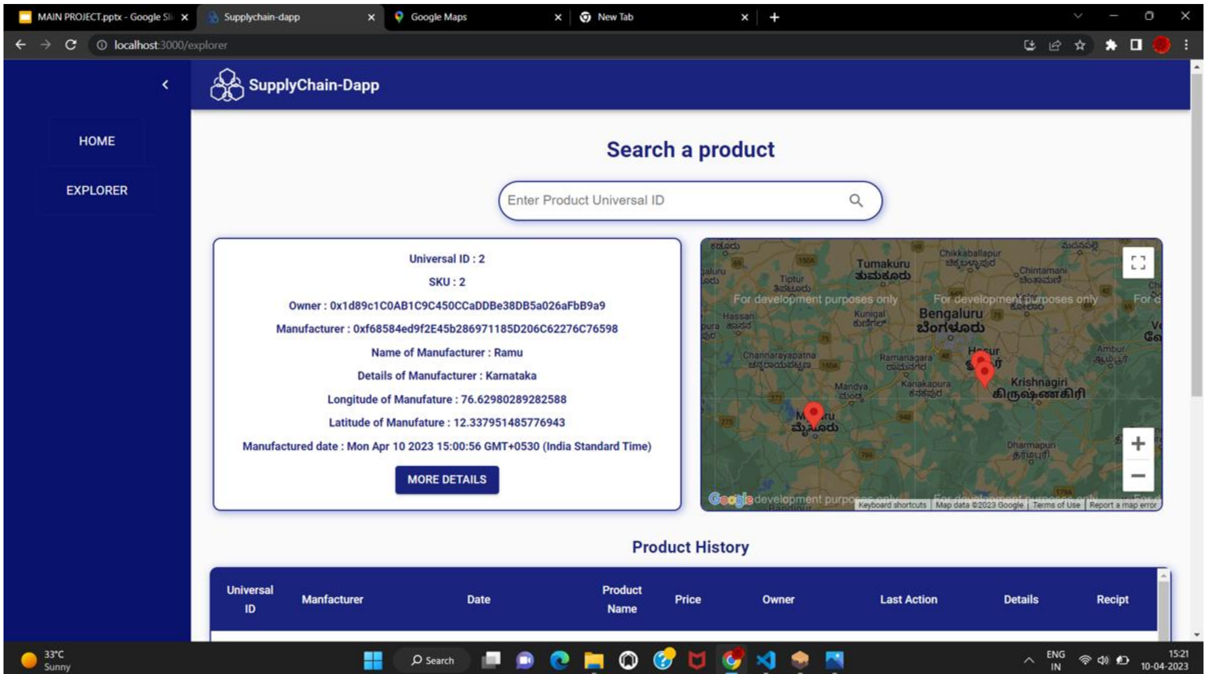


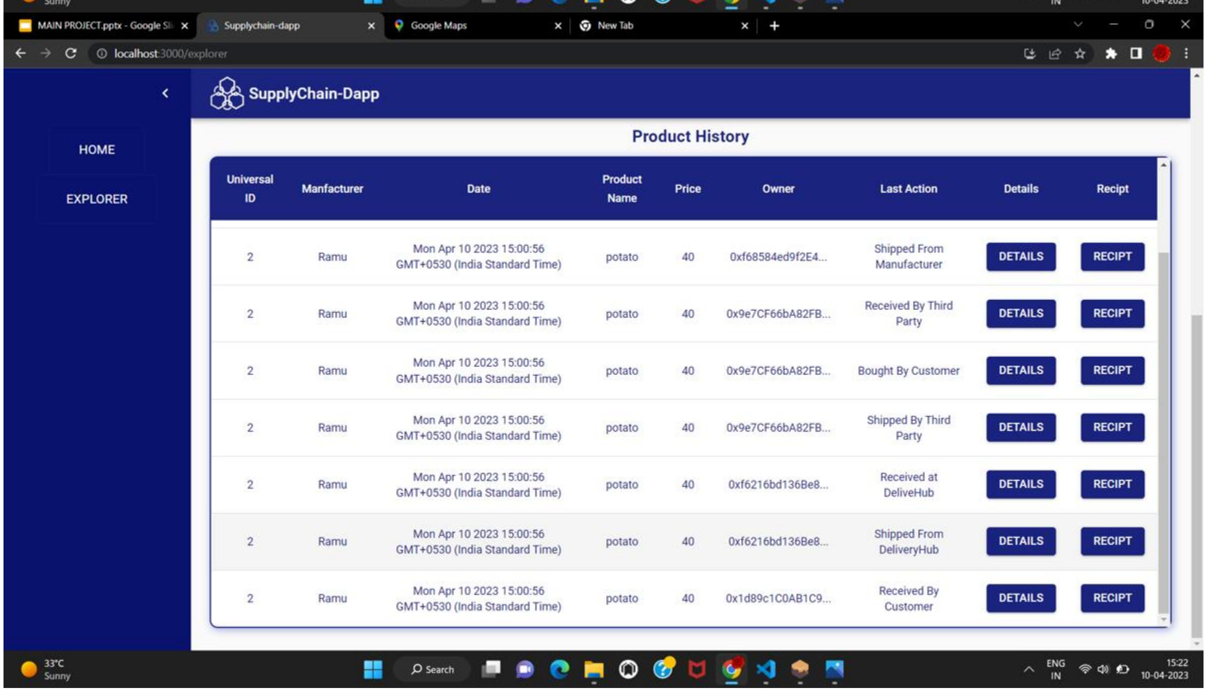

localhost:3000/manufacturer/manufacture
Manufacturer
SupplyChain-Dapp
HOME
EXPLORER
Add Product
Ship Product
All Products
Add Product
Manufacturer Name *
Ramu
Manufacturer Details *
Karnataka
Longitude *
76.62980289282588
Latitude *
12.337951485776943
Product Name *
potato
Product Code *
2
Product Price *
40
Product Category *
vegetable
SUBMIT


localhost:3000/manufacturer/allManufacture
Manufacturer
SupplyChain-Dapp
HOME
EXPLORER
Add Product
Ship Product
All Products
Manufactured Products
Total : 7
Universal ID | Product Code | Manufacturer | Manufacture Date | Product Name | Owner
2 | 2 | Ramu | Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) | potato | 0xf68584ed9f2E45...
3 | 3 | Priya | Mon Apr 10 2023 15:02:58 GMT+0530 (India Standard Time) | Green chilli | 0xf68584ed9f2E45...
4 | 4 | Dharshini | Mon Apr 10 2023 15:04:40 GMT+0530 (India Standard Time) | coconut | 0xf68584ed9f2E45...
5 | 5 | Indhirani | Mon Apr 10 2023 15:06:11 GMT+0530 (India Standard Time) | grapes | 0xf68584ed9f2E45...
6 | 6 | Jayapriya | Mon Apr 10 2023 15:08:15 GMT+0530 (India Standard Time) | strawberry | 0xf68584ed9f2E45...
7 | 7 | Meena | Mon Apr 10 2023 15:10:17 GMT+0530 (India Standard Time) | lady's finger | 0xf68584ed9f2E45...
8 | 8 | Varshini | Mon Apr 10 2023 15:11:47 GMT+0530 (India Standard Time) | beetroot | 0xf68584ed9f2E45...
Rows per page: 10
1-7 of 7


MAIN PROJECT.pptx - Google Sli
Supplychain-dapp
Google Maps
New Tab
localhost:3000/manufacturer/ship
Manufacturer
SupplyChain-Dapp
HOME
EXPLORER
Add Product
Ship Product
All Products
Products To be Shipped
Total : 1
Universal ID
Product Code
Manufacturer
Manufacture Date
Product Name
Owner
Ship
2
2
Ramu
Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time)
potato
0xf68584ed9f2E4...
SHIP
Rows per page: 10
1-1 of 1
33°C
Sunny
Search
ENG
IN
15:13
10-04-2023


MAIN PROJECT.pptx - Google Sli
Supplychain-dapp
Google Maps
New Tab
localhost:3000/Customer/allReceived
Customer
SupplyChain-Dapp
HOME
EXPLORER
Purchase Product
Receive Product
Your Products
Your Products
Total : 2
Universal ID
Product Code
Manufacturer
Manufacture Date
Product Name
Owner
1
1
Jaya
Mon Apr 10 2023 11:23:55 GMT+0530 (India Standard Time)
apple
0x1d89c1C0AB1C9...
2
2
Ramu
Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time)
potato
0x1d89c1C0AB1C9...
Rows per page: 10
1-2 of 2
33°C
Sunny
Search
ENG
IN
15:20
10-04-2023


SupplyChain-Dapp
HOME
EXPLORER
Search a product
Enter Product Universal ID
Universal ID : 2
SKU : 2
Owner : 0x1d89c1C0AB1C9C450CCaDDBe38DB5a026aFbB9a9
Manufacturer : 0xf68584ed9f2E45b286971185D206C62276C76598
Name of Manufacturer : Ramu
Details of Manufacturer : Karnataka
Longitude of Manufature : 76.62980289282588
Latitude of Manufature : 12.337951485776943
Manufactured date : Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time)
MORE DETAILS
Tumakuru
Bengaluru
Krishnagiri
For development purposes only
Product History
Universal ID
Manfacturer
Date
Product Name
Price
Owner
Last Action
Details
Recipt


SupplyChain-Dapp
HOME
EXPLORER
Product History
Universal ID
Manfacturer
Date
Product Name
Price
Owner
Last Action
Details
Recipt
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0xf68584ed9f2E4... Shipped From Manufacturer DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0x9e7CF66bA82FB... Received By Third Party DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0x9e7CF66bA82FB... Bought By Customer DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0x9e7CF66bA82FB... Shipped By Third Party DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0xf6216bd136Be8... Received at DeliveHub DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0xf6216bd136Be8... Shipped From DeliveryHub DETAILS RECIPT
2 Ramu Mon Apr 10 2023 15:00:56 GMT+0530 (India Standard Time) potato 40 0x1d89c1C0AB1C9... Received By Customer DETAILS RECIPT

CHAPTER 6
CONCLUSION

6.1 CONCLUSION

The traditional traceability systems have issues such as data invisibility, tampering and sensitive information disclosure. The traceability based on blockchain allows creating a decentralized and immutable ledger of transactions which are verifiable and traceable.

The proposed method, which is based on blockchain enhances transparency. It is a useful tracking tool throughout the supply chain lifecycle. As a food product meets different checkpoints, its data can be documented and updated on blockchain ledgers, so users can view every step of the product’s journey.

CHAPTER 7
REFERENCES

7.1 REFERENCES

[1] KHALED SALAH, 2019. “Blockchain Based Soyabean Traceability In Agricultural Supply Chain”, IEEE.

[2] SHANGPING WANG, 2019, “Smart Contract-Based Product Traceability System in the Supply Chain Scenario”, IEEE.

[3] PRATYUSH KUMAR PATRO, 2022, “Blockchain-Based Traceability for the Fishery Supply Chain”, IEEE.

[4] XIN ZHANG, 2020, “Blockchain-Based Safety Management System for the Grain Supply Chain”, IEEE.

[5] LUISANNA COCCO, 2022, “Blockchain-Based Traceability System in Agri-Food SME: Case Study of a Traditional Bakery”, IEEE.

[6] SATIT KRAVENKIT, 2022, “Blockchain-Based Traceability System for Product Recall”, IEEE.

[7] AFFAF SHAHID, 2020, “Blockchain-Based Agri-Food Supply Chain: A Complete Solution”, IEEE.