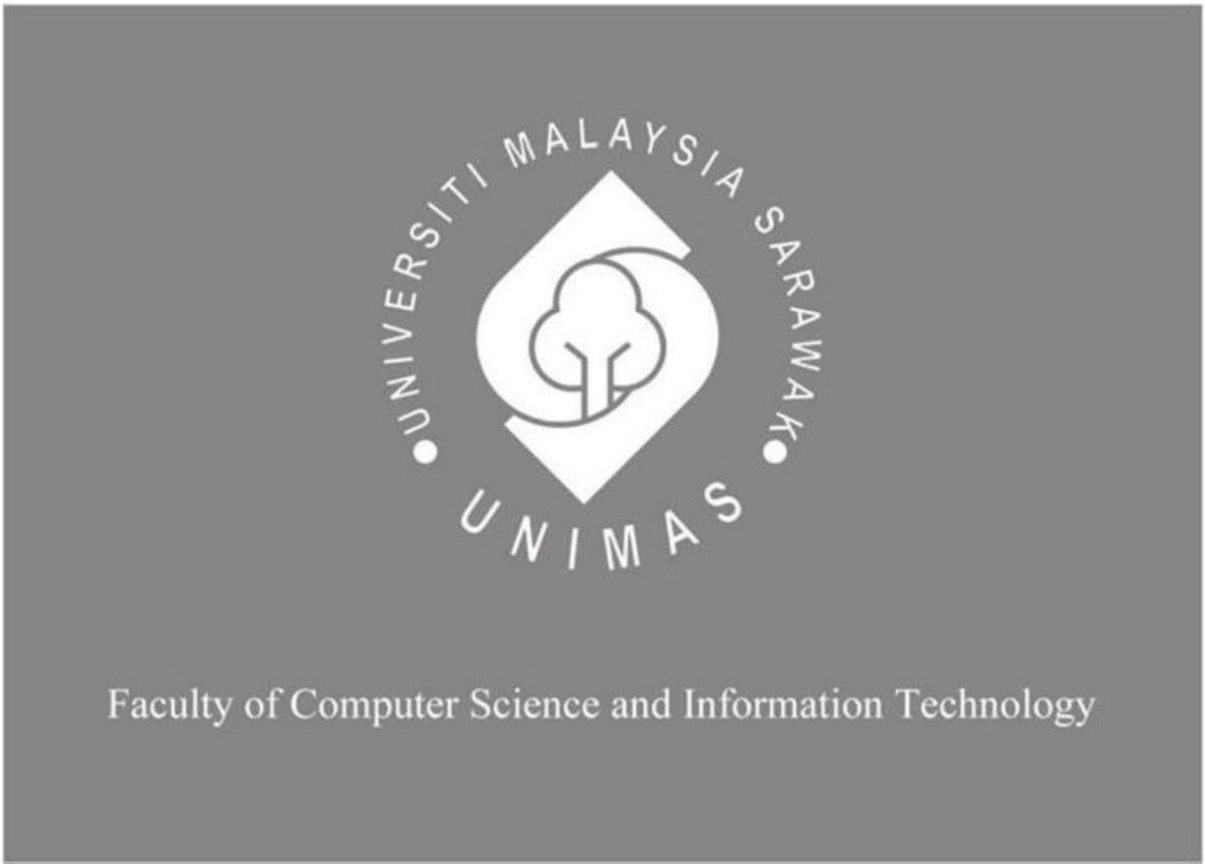


A BLOCKCHAIN SYSTEM FOR FOOD PRODUCT TOKENIZATION AND CONSUMER VERIFICATION IN FOOD SUPPLY INDUSTRY

KENG WEI SHENG

Bachelor of Software Engineering with Honors

2024

A BLOCKCHAIN SYSTEM FOR FOOD PRODUCT TOKENIZATION AND CONSUMER VERIFICATION IN FOOD SUPPLY INDUSTRY

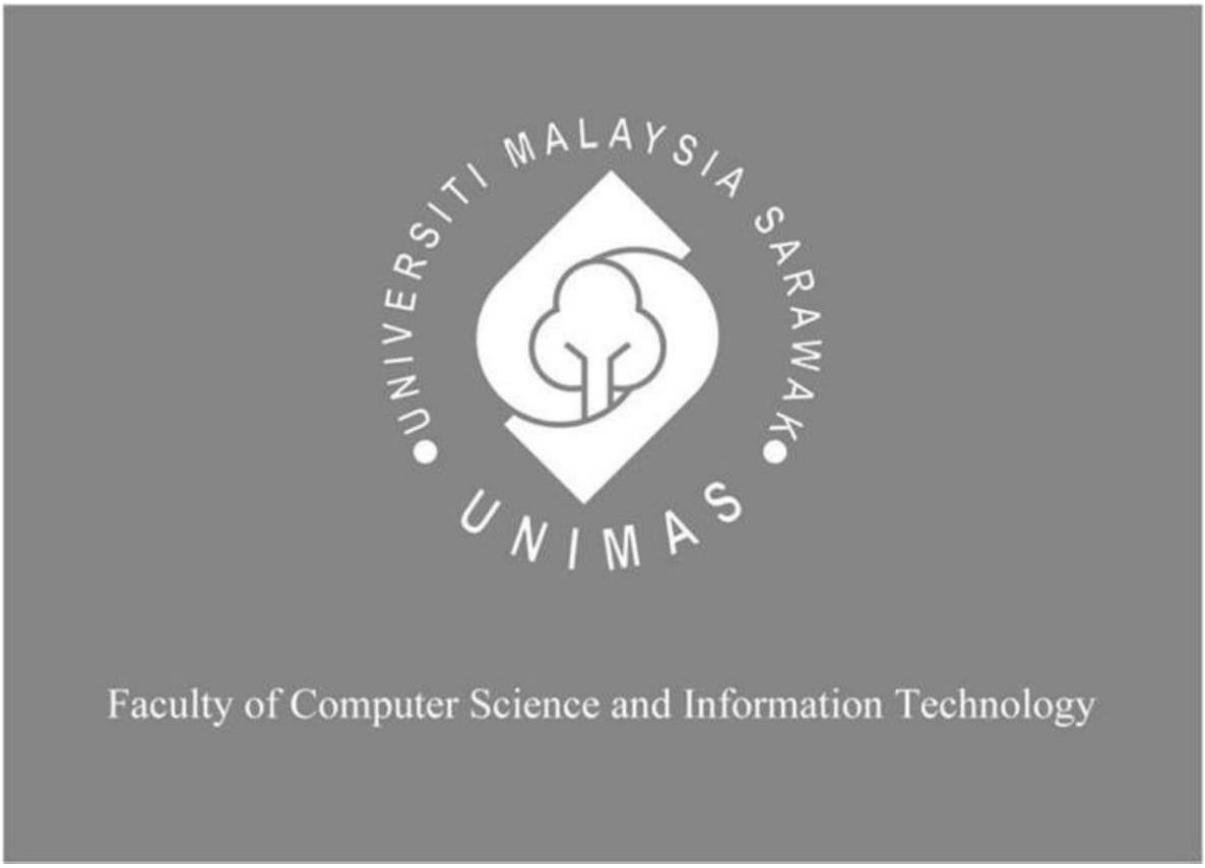
KENG WEI SHENG

This project is submitted in partial fulfilment of

the requirements for the degree of Bachelor of Software Engineering with

Honors

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2024



SISTEM BLOCKCHAIN UNTUK PENGETOKAN PRODUK MAKANAN DAN PENGESAHAN PENGGUNA DALAM INDUSTRI RANTAIAN BEKALAN MAKANAN

KENG WEI SHENG

Ijazah Sarjana Muda Kejuruteraan Perisian dengan Kepujian

2024

SISTEM BLOCKCHAIN UNTUK PENGETOKAN PRODUK MAKANAN DAN PENGESAHAN PENGGUNA DALAM INDUSTRI RANTAIAN BEKALAN MAKANAN

KENG WEI SHENG

Projek ini merupakan salah satu keperluan untuk

Ijazah Sarjana Muda Kejuruteraan Perisian dengan Kepujian

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2024

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE A BLOCKCHAIN SYSTEM FOR FOOD PRODUCT TOKENIZATION AND CONSUMER VERIFICATION IN FOOD SUPPLY INDUSTRY

ACADEMIC SESSION: 2024/2025

KENG WEI SHENG

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
☑	UNRESTRICTED	

Validated by

(AUTHOR'S SIGNATURE)

(SUPERVISOR'S SIGNATURE)

Permanent Address

No 7633, Taman Rakyat,
Jalan Parit Semerah,
82000 Pontian, Johor

Date: 10 June 2025

Date: _____

Note * Thesis refers to PhD, Master, and Bachelor Degree

** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

UNIVERSITI MALAYSIA SARAWAK

Grade: _____

Please tick (√)	
Final Year Project Report	√
Masters	
PhD	

DECLARATION OF ORIGINAL WORK

This declaration is made on the 10th day of July (2025).

Student's Declaration:

I Keng Wei Sheng, 79761, Faculty of Computer Science and Information Technology hereby declare that the work entitled, A Blockchain System For Food Product Tokenization And Consumer Verification In Food Supply Industry is my original work. I have not copied from any other students' work or from any other sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

10 June 2025
Date submitted

KENG WEI SHENG (79761)
Name of the student (Matric No.)

Supervisor's Declaration:

I Wang Yin Chai (SUPERVISOR'S NAME) hereby certifies that the work entitled, A Blockchain System For Food Product Tokenization And Consumer Verification In Food Supply Industry was prepared by the above named student, and was submitted to the "FACULTY" as a * partial/full fulfillment for the conferment of Bachelor of Software Engineering with Honors, and the aforementioned work, to the best of my knowledge, is the said student's work

Received for examination by: Wang Yin Chai Date: _____
(Name of the supervisor)

I declare this Project/Thesis is classified as (Please tick (√)):

☐ **CONFIDENTIAL** (Contains confidential information under the Official Secret Act 1972)*

☐ **RESTRICTED** (Contains restricted information as specified by the organisation where research was done)*

☑ **OPEN ACCESS**

Validation of Project/Thesis

I therefore duly affirmed with free consent and willingness declared that this said Project/Thesis shall be placed officially in the Centre for Academic Information Services with the abide interest and rights as follows:

- This Project/Thesis is the sole legal property of Universiti Malaysia Sarawak (UNIMAS).
- The Centre for Academic Information Services has the lawful right to make copies for the purpose of academic and research only and not for other purpose.
- The Centre for Academic Information Services has the lawful right to digitise the content to for the Local Content Database.
- The Centre for Academic Information Services has the lawful right to make copies of the Project/Thesis for academic exchange between Higher Learning Institute.
- No dispute or any claim shall arise from the student itself neither third party on this Project/Thesis once it becomes sole property of UNIMAS.
- This Project/Thesis or any material, data and information related to it shall not be distributed, published or disclosed to any party by the student except with UNIMAS permission.

Student's signature  Supervisor's signature: _____

(10/07/2025) (Date)

Current Address:
No 7633, Taman Rakyat, Jalan Parit Semerah, 82000 Pontian, Johor.

Notes: * If the Project/Thesis is **CONFIDENTIAL** or **RESTRICTED**, please attach together as an annexure a letter from the organisation with the period and reasons of confidentiality and restriction.

[The instrument was duly prepared by The Centre for Academic Information Services]

ACKNOWLEDGEMENT

I would like to extend my heartfelt gratitude to my supervisor, Professor Dr. Wang Yin Chai, for his invaluable guidance and unwavering support throughout the completion of this project. His insight, dedication, and encouragement have greatly inspired me, and it has been an honour and privilege to work under his mentorship.

I am also deeply thankful to my examiner, Associate Professor Ts. Dr. Adnan Shahid Khan, for his constructive advice and insightful feedback, which have significantly enhanced the quality of this project.

Additionally, I would like to express my appreciation to Universiti Malaysia Sarawak (UNIMAS) and the Faculty of Computer Science and Information Technology (FCSIT) for providing me with the opportunity to undertake this final year project. This experience has allowed me to explore and acquire valuable knowledge and skills.

I am also grateful to everyone who contributed to the requirement-gathering process and survey participation. Their input was crucial in helping me gather the necessary data and feedback to develop this project effectively.

Lastly, I wish to thank my beloved family, supportive friends, and dedicated coursemates for their continuous encouragement and moral support. Their presence and motivation helped me navigate through the challenging and stressful moments of this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

As consumer pressure for transparency and accountability in the food sector rises, blockchain technology has surfaced as a viable means to tackle issues related to data security, integrity, and traceability within the food supply chain. Conventional centralized systems frequently face risks of data manipulation and inefficiencies in handling refunds and compensation processes. This initiative is centred on creating a blockchain-based framework that enhances the consumer experience by granting immediate access to product details, incorporating QR codes for greater transparency, and using product tokenization to verify ownership. By implementing smart contracts, the framework seeks to automate the processes for refunds and compensation, thereby optimizing operations and minimizing processing timelines. The suggested solution utilizes Ethereum-based technologies including Hardhat, IPFS, Solidity, and ERC1155 NFTs to establish a decentralized, secure, and tamper-proof platform for consumers to interact with the food supply chain. By increasing transparency and cutting down on inefficiencies, this project adds value to both academic research and practical use cases of blockchain in consumer-oriented sectors, presenting a scalable framework for enhancing trust, security, and efficiency in food supply chains.

ABSTRAK

Dengan peningkatan tekanan daripada pengguna untuk mendapatkan ketelusan dan akauntabiliti dalam sektor makanan, teknologi blockchain telah muncul sebagai kaedah yang berpotensi untuk menangani isu berkaitan keselamatan data, integriti, dan kebolehkesanan dalam rantaian bekalan makanan. Sistem berpusat konvensional menghadapi risiko manipulasi data dan ketidakcekapan dalam mengendalikan proses pemulangan wang dan pampasan. Inisiatif ini bertujuan untuk membangunkan rangka kerja berasaskan blockchain yang meningkatkan pengalaman pengguna dengan memberikan akses segera kepada butiran produk, mengintegrasikan kod QR untuk ketelusan yang lebih baik, dan menggunakan tokenisasi produk untuk mengesahkan pemilikan. Melalui pelaksanaan kontrak pintar, rangka kerja ini bertujuan untuk mengautomasikan proses pemulangan wang dan pampasan, sekali gus mengoptimumkan operasi dan mengurangkan tempoh pemprosesan. Penyelesaian yang dicadangkan menggunakan teknologi berasaskan Ethereum seperti Hardhat, IPFS, Solidity, dan NFT ERC1155 untuk membentuk platform terdesentralisasi, selamat, dan bebas manipulasi bagi pengguna untuk berinteraksi dengan rantaian bekalan makanan. Dengan meningkatkan ketelusan dan mengurangkan ketidakcekapan, projek ini memberikan nilai tambah kepada penyelidikan akademik dan aplikasi praktikal blockchain dalam sektor berorientasikan pengguna, serta membentangkan rangka kerja berskala untuk meningkatkan kepercayaan, keselamatan, dan kecekapan dalam rantaian bekalan makanan.

CHAPTER 1 INTRODUCTION

1.1 Background

In recent years, consumers have shown an increasing demand for transparency and accountability in the products they purchase, particularly in the food industry (Rejeb, Keogh, Zailani, Treiblmaier, & Rejeb, 2020). This trend has been fuelled by a growing awareness of food safety, ethical sourcing, and sustainability, leading consumers to seek assurance that the products they buy meet certain quality standards. However, traditional centralized systems in the food supply chain often struggle to meet these demands due to challenges in data integrity, security, and transparency (Iftekhar, Cui, Hassan, & Afzal, 2020). These centralized systems can be prone to data tampering and unauthorized access, making it difficult for consumers to fully trust the information they receive about a product's origin, handling, and certifications.

Blockchain technology has emerged as a promising solution to address these issues by providing a decentralized, tamper-resistant way of storing and sharing data. As a distributed ledger technology, blockchain allows for secure, transparent record-keeping across multiple stakeholders without the need for a central authority, which can greatly benefit supply chain management by ensuring that data is accurate, up-to-date, and accessible to all relevant parties (Tian, 2016). In the food industry, blockchain's capabilities can help verify product authenticity, enhance traceability, and ensure that quality certifications are readily available to consumers. Blockchain-based systems allow each transaction to be securely stored and timestamped, creating a traceable product journey that increases accountability throughout the supply chain (Kshetri, 2017).

Despite its potential, implementing blockchain in the food supply chain remains complex, especially in integrating data from various sources and automating interactions between stakeholders. Many existing studies on blockchain in the supply chain have focused

on its benefits for producers and suppliers, leaving consumer-side applications relatively underexplored. The consumer-facing features such as the ability to scan a product's QR code for detailed information on its origins and certifications could significantly enhance transparency and trust (Galvez, Mejuto, & Simal-Gandara, 2018). This project seeks to address this gap by developing a blockchain-based system that focuses specifically on the consumer experience, enabling users to access reliable product information and interact with the supply chain through features such as product tokenization, automated ownership transfer, and smart contract-based compensation processes.

The proposed system aims to tackle inefficiencies in current refund and compensation procedures within the food industry. Traditionally, these processes require manual data verification among multiple stakeholders, resulting in prolonged processing times, increased communication costs, and a high likelihood of errors. These inefficiencies not only damage consumer trust but also affect the brand's reputation and overall consumer satisfaction (Rejeb et al., 2020). By implementing blockchain and smart contract automation, this project intends to streamline these processes, reduce operational costs, and provide a more reliable method for addressing consumer complaints and compensations.

Overall, this project is positioned at the intersection of consumer demand for transparency and blockchain's capabilities in providing secure, traceable data storage. By developing a consumer-focused blockchain application for the food supply chain, the project aims to contribute to both research and practical applications of blockchain technology, demonstrating its effectiveness in enhancing transparency, security, and efficiency in consumer-facing sectors of the supply chain.

1.2 Problem Statement

In recent years, consumers are increasingly demanding greater transparency and accountability in the products they purchase (Rejeb et al., 2020) . However, the traditional centralized systems currently used in the food supply industry struggle to meet these demands. These systems are vulnerable to data tampering and loss, leading to concerns about data integrity, security, and transparency (Iftekhar et al., 2020).

Additionally, the refund or compensation process in the food supply industry is complex and involves multiple stakeholders. Each party requires manual verification of data, leading to prolonged processing times, increased communication costs, and a higher likelihood of errors. These inefficiencies not only damage consumer trust but also undermine confidence in the safety and quality of food products. The reliance on fragmented, non-integrated systems exacerbates these challenges, making it difficult for consumers to receive timely and transparent updates, further eroding their trust in the process.

1.3 Objectives

This project aims to develop the blockchain-based system which is able:

1. To provide consumers with real-time access to product information and related certifications by using QR codes
2. To implement product tokenization and automatic transfer of ownership to ensure that each consumer can be verified when purchasing a product
3. To utilize smart contracts to automate the handling of refund and compensation requests.

1.4 Methodology

This project employs a structured approach to achieve its objectives by utilizing blockchain technology to create a decentralized, secure, and transparent system that addresses

challenges in the food supply chain. Key steps include designing and deploying blockchain-based smart contracts, integrating QR code technology, tokenizing products for ownership tracking, and building a user-friendly interface. Essential tools and techniques for this process include the Hardhat blockchain framework for development, IPFS for decentralized storage, Solidity for smart contract coding, and ERC1155 NFT standards for tokenization. These components will support the system's core functionalities of transparency, security, and efficient process automation.

Hardhat will serve as the primary blockchain framework to ensure a reliable development and testing environment. Hardhat is specifically designed for Ethereum, allowing for rapid smart contract testing, debugging, and deployment. Additionally, the system will utilize IPFS (InterPlanetary File System) for decentralized data storage. IPFS will store critical product-related documents and certifications in a tamper-proof way, maintaining data integrity and accessibility for consumers seeking supply chain information. By using IPFS alongside Hardhat, the project can create a secure and decentralized environment where essential information remains verifiable and transparent.

The smart contracts, coded in Solidity, will handle key functions in the blockchain system, such as automated refunds, compensation processes, and product ownership transfers. Solidity, as Ethereum's primary language, is optimized for creating reliable, tamper-resistant contracts that execute automatically based on predefined conditions. Testing of these contracts within Hardhat will simulate various scenarios, ensuring they work seamlessly in automating critical supply chain processes. Solidity's flexibility also supports the customization of smart contracts to handle specific conditions.

Product tokenization will be implemented through the ERC1155 NFT (non-fungible token) standard, which will enable each product to have a unique digital representation on the

blockchain. ERC1155 tokens are particularly suited to this purpose because they can securely verify and track ownership through blockchain records. Each tokenized product will also have a unique QR code linked to its blockchain entry, allowing consumers to access real-time information by simply scanning the code. This integration of NFTs and QR codes will provide a streamlined, reliable method for tracing product history, ownership, and quality certifications.

For front-end development, HTML, CSS, and Ether.js will be used to build a responsive web interface where consumers can interact with the blockchain-based system. Ether.js, a JavaScript library for interacting with the Ethereum blockchain, will facilitate communication between the front end and the blockchain, allowing users to view product details, verify ownership transfers, and scan QR codes to access product information. This setup will provide consumers with an accessible and efficient platform for engaging with the system, ensuring a seamless user experience across devices.

1.5 Scope

This system is developed using Web 3.0 technologies, with a specific focus on decentralization, transparency, and data security. By implementing blockchain, the project aims to address consumer demands for transparency and accountability within the food supply chain. The system will allow consumers to access product information through QR codes, thereby enhancing their trust in food products. However, the project does not cover data input by suppliers, meaning that only consumer-side interactions with the blockchain are included.

This project specifically focuses on solving issues related to consumer access to product information, which includes viewing certifications, traceability, and transaction history. The primary problem addressed is the lack of consumer trust due to insufficient transparency in the traditional centralized food supply chain. However, it does not address all supply chain issues,

particularly those related to supplier-side data management, production tracking, or complex distribution networks.

The scope of this project is also limited to a simulated food supply chain environment rather than an actual, complex commercial supply chain. The system will model a simplified version of the food supply chain for demonstration purposes, allowing consumers to interact with a blockchain-based system without involving the extensive and variable data sources found in real-world scenarios.

In terms of functionality, the project will cover consumer-side features such as QR code scanning, product information retrieval, and automated smart contract handling of refunds or compensation. It will not include supplier functions like product data uploads, manufacturing logs, or logistical updates. This boundary keeps the project manageable and emphasizes the consumer experience while excluding supplier responsibilities.

1.6 Significant of Project

The significant of this project is applying blockchain technology to solve real-world problems in the food supply chain. Blockchain, as a decentralized and tamper-resistant technology, ensures transparency, security, and accountability, which are critical in sectors that rely on data integrity. The development of a blockchain-based system for product traceability and consumer interaction introduces new methods for secure data handling and automated processes, which are significant contributions to the field of computer science. Furthermore, this project explores the use of Web 3.0 technologies, such as smart contracts and decentralized storage, providing valuable insights into how these technologies can be applied in everyday consumer-facing applications.

This project also touches on multiple areas, including blockchain technology, data privacy, decentralized systems, and smart contracts, offering valuable insights for both academic and industry applications. By implementing a secure, transparent, and tamper-resistant system, it demonstrates the potential of blockchain to enhance consumer trust and accountability in the food supply chain. The findings can highlight how blockchain can be applied in similar consumer-driven industries to improve transparency, security, and process efficiency, showcasing blockchain's practicality and adaptability beyond traditional finance applications.

1.7 Expected Outcome

The expected outcomes of this project are focus on improving transparency, security, and efficiency in the food supply chain. First, the system is expected to enhance consumer transparency by providing real-time access to verified product information through QR codes. Consumers will be able to view the origins, certifications, and details of products, helping to build trust and empower them to make more informed purchasing decisions. Additionally, by utilizing ERC1155 NFT tokenization, each product will have a unique digital identity on the blockchain, enabling reliable ownership tracking. This feature will create a secure method for verifying product authenticity and ownership history, which is particularly valuable for high-value or sensitive food items.

Furthermore, the project will improve the efficiency of refund and compensation processes. Smart contracts will automate these processes, reducing manual verification efforts and minimizing processing time. This automation is anticipated to streamline consumer transactions, lower operational costs, and increase user satisfaction by ensuring quick and fair resolutions. Data security and integrity are also key outcomes, as the decentralized design will rely on IPFS for data storage, making information tamper-resistant and reducing risks of data loss or unauthorized access. This will assure consumers that product data remains accurate and secure.

Overall, the project is expected to offer a scalable model demonstrating blockchain's feasibility in consumer-facing supply chains, providing a framework for potential applications in other industries where transparency and traceability are essential. By establishing practical use cases and workflows, this project can guide and inspire further adoption of blockchain technology, especially in areas where consumer trust, product authenticity, and secure data sharing are vital concerns.

1.8 Project Outlines

This project is organized into five chapters. Each chapter addressing a specific aspect of the development process and research related to the blockchain-based system for the food supply chain.

The first chapter is Introduction. It introduces the project by highlighting the motivation behind its development and identifying the key issues faced by traditional centralized systems in the food supply chain. It also includes the problem statement, objectives, scope, and significance to emphasize the consumer-focused approach to enhance transparency, security, and efficiency using blockchain technology.

The next chapter is the Background Study and Literature Review. This chapter provides a comprehensive review of existing literature and related work. It examines the application of blockchain in food supply chain management and discusses key technologies such as smart contracts, ERC1155 NFTs, and QR codes. The chapter also highlights the benefits of blockchain for data integrity and transparency.

The third chapter, Requirement Analysis and Design, focuses on the technical foundation of the system. It outlines the system requirements, including functional requirements and non-functional requirements. The chapter presents the proposed system architecture, detailing the blockchain framework, smart contract design, and the integration of IPFS for decentralized data storage.

Furthermore, the fourth chapter is the Implementation and Testing. This chapter documents the development process of the system. It describes the implementation of the blockchain-based solution, including the creation of smart contracts in Solidity and the front-end interface using HTML, CSS, Web3.js, and Bootstrap. Testing methods and results are

presented to evaluate the system's performance and reliability to ensure that the developed solution aligns with the project objectives.

Conclusion is the last chapter in the project**.** It concludes the project by summarizing its achievements and evaluating its contributions to addressing the identified challenges. It also discusses the significance of the developed system, reflecting on its impact on consumer trust and the food supply chain's transparency.

This structured approach ensures that the project is comprehensively documented, from problem identification to the delivery of a functional blockchain-based system, while highlighting its contribution to the food supply chain and blockchain research.

1.9 Summary

Chapter 1, Introduction, introduces the project by outlining the problem statement, objectives, scope, and significance of the proposed blockchain-based system for the food supply chain. It identifies the limitations of traditional centralized systems, such as data tampering, inefficiencies in refund processes, and lack of transparency, which undermine consumer trust. The chapter highlights blockchain technology as a potential solution, emphasizing its ability to provide decentralized, secure, and transparent data management.

The project's focus is on the consumer side, aiming to enhance transparency and accountability by enabling real-time access to product information through QR codes, automating ownership transfer via product tokenization, and streamlining refund processes using smart contracts. The scope of the project is limited to a simulated food supply chain environment and does not involve the supplier-side data upload process.

Lastly, the chapter underscores the significance of the project in contributing to computer science by exploring blockchain's application in the consumer-facing segment of the

food supply chain, addressing challenges in data integrity, and improving user trust and satisfaction.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

This chapter presents a literature review of research papers related to blockchain systems in food supply chain management. The selected papers are reviewed and analysed based on key aspects such as transparency, traceability, scalability, tokenization, smart contracts, QR code integration, IPFS integration, automated refund, and their identified weaknesses. This review aims to provide insights and identify best practices to inform the development of the proposed system.

2.2 Blockchain-based Traceability in Agri-Food Supply Chain Management: A Practical Implementation

The paper focuses on AgriBlockIoT, a blockchain-based traceability solution designed for Agri-Food supply chain management. It integrates IoT devices with blockchain technology to ensure transparency, traceability, and the immutability of records. In terms of scalability, the paper compares Ethereum and Hyperledger Sawtooth as blockchain implementations. Ethereum offers high scalability due to its maturity and capacity to support a large number of participants but suffers from higher latency. On the other hand, Hyperledger Sawtooth demonstrates lower latency and resource requirements but is less mature and widely adopted. Security is ensured through the blockchain's tamper-proof and auditable records, with public/private key pairs enhancing secure participation. Transparency is a key feature, allowing participants to access the complete history of products throughout the supply chain.

The paper mentions the use of QR codes to automate processes such as recording raw material purchases and enabling consumers to retrieve detailed product information. Smart contracts are also applied in the system, automating tasks like anomaly detection and ensuring compliance with regulations. However, tokenization is not discussed in the paper. Some

weaknesses identified include Ethereum's high latency and CPU-intensive consensus algorithm and Hyperledger Sawtooth's lack of maturity and limited adoption. Additionally, integrating IoT devices into blockchain systems poses challenges due to varying hardware constraints.

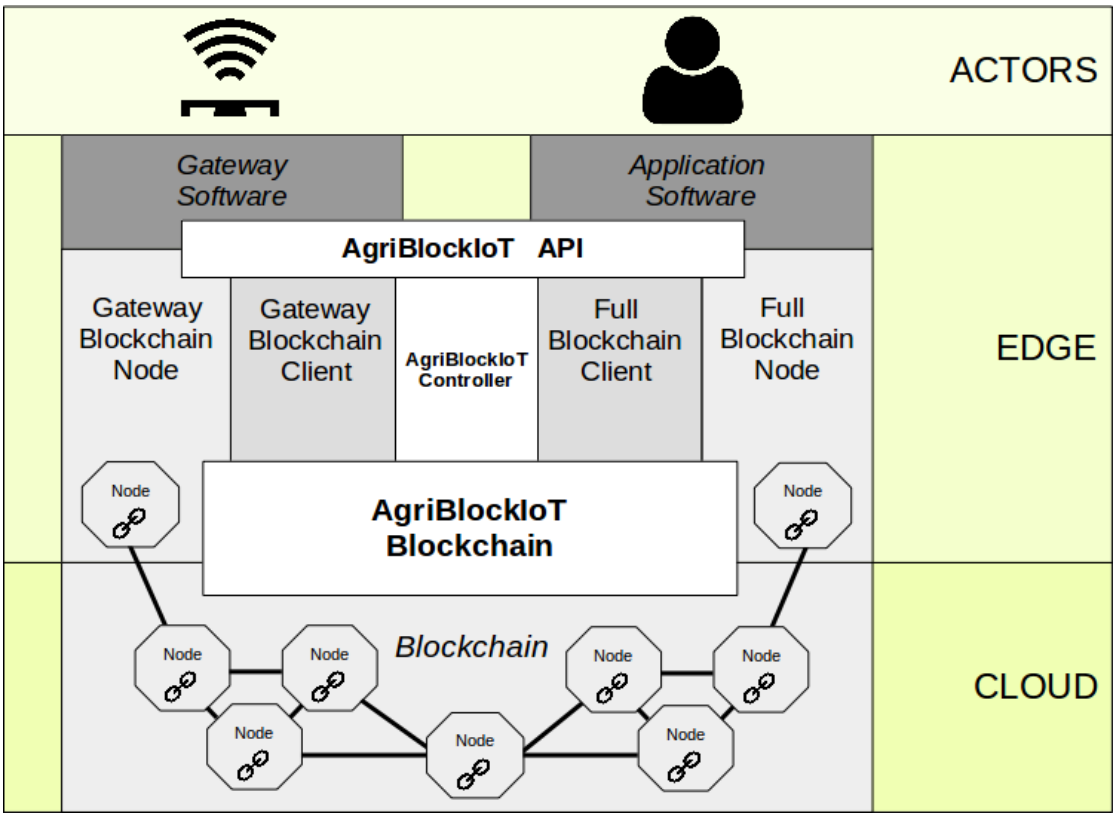


Figure 2.1: Layered architecture of AgriBlockIoT (Caro et al., 2018).

2.3 Blockchain-based Agri-food Supply Chain: A Complete Solution

The paper provides a study of an agri-food supply chain solution that is based on blockchain technology. The solution aims to overcome the challenges regarding transparency, traceability, and credibility that conventional supply chains face. Information within a supply chain management system is often stored on a single centralized server, thus making the system's accuracy and transparency highly susceptible to fraud or tampering. The suggested system in this paper is built on the Ethereum blockchain and uses smart contracts, decentralized storage (IPFS), and reputation mechanisms to make the supply chain more open, able to be

tracked, and safe for everyone involved. This makes the model of supply chain management safer, more reliable, and more effective.

The proposed system in this study has several beneficial aspects. First, with blockchain technology, the system guarantees immutability and transparency of all transactions. Every transaction is recorded on a blockchain, hence avoiding the risk of data loss or manipulation that exists in traditional centralized systems. In addition, the incorporation of IPFS as a decentralized storage option allows the system to solve the problems associated with large volumes of data and the storage bottlenecks present in conventional systems. The opacity of the system has also been drastically reduced. The combination of blockchain and smart contracts permits the monitoring of every unit of the product from its production to the final consumer, thus guaranteeing product quality traceability. Moreover, the incorporation of smart contracts enables greater automation of the transaction process, less human interaction, and greater stakeholder protection, thus allowing for dependable payments and effective dispute resolution. Last, but not least, the system implements a reputation system that adds to the trust of the participants. Consumers can assess the credibility of the seller based on past transactions, thus enabling them to make purchase decisions with greater confidence.

Although the system proposed in this paper has various impressive points, there are some weaknesses too. First, the paper under review does not tend to elaborate deeply on the issue of system tokenization. Though there is provision of tokenization within the blockchain, there is none in terms of how to carry out product transactions or supply chain improvement through tokenization. Secondly, while the paper does address the use of smart contracts for automating transactions and payment mechanisms, there is no detail provided on how the refund mechanism is embraced especially in terms of automatic refunds during dispute scenarios. Lastly, although the author suggests the implementation of a reputation management system,

there are still concerns about how the system's overall credibility may be impacted by false reviews or ratings. Overall, more work is needed in the system tokenization, refund process, and protection against malicious actions.

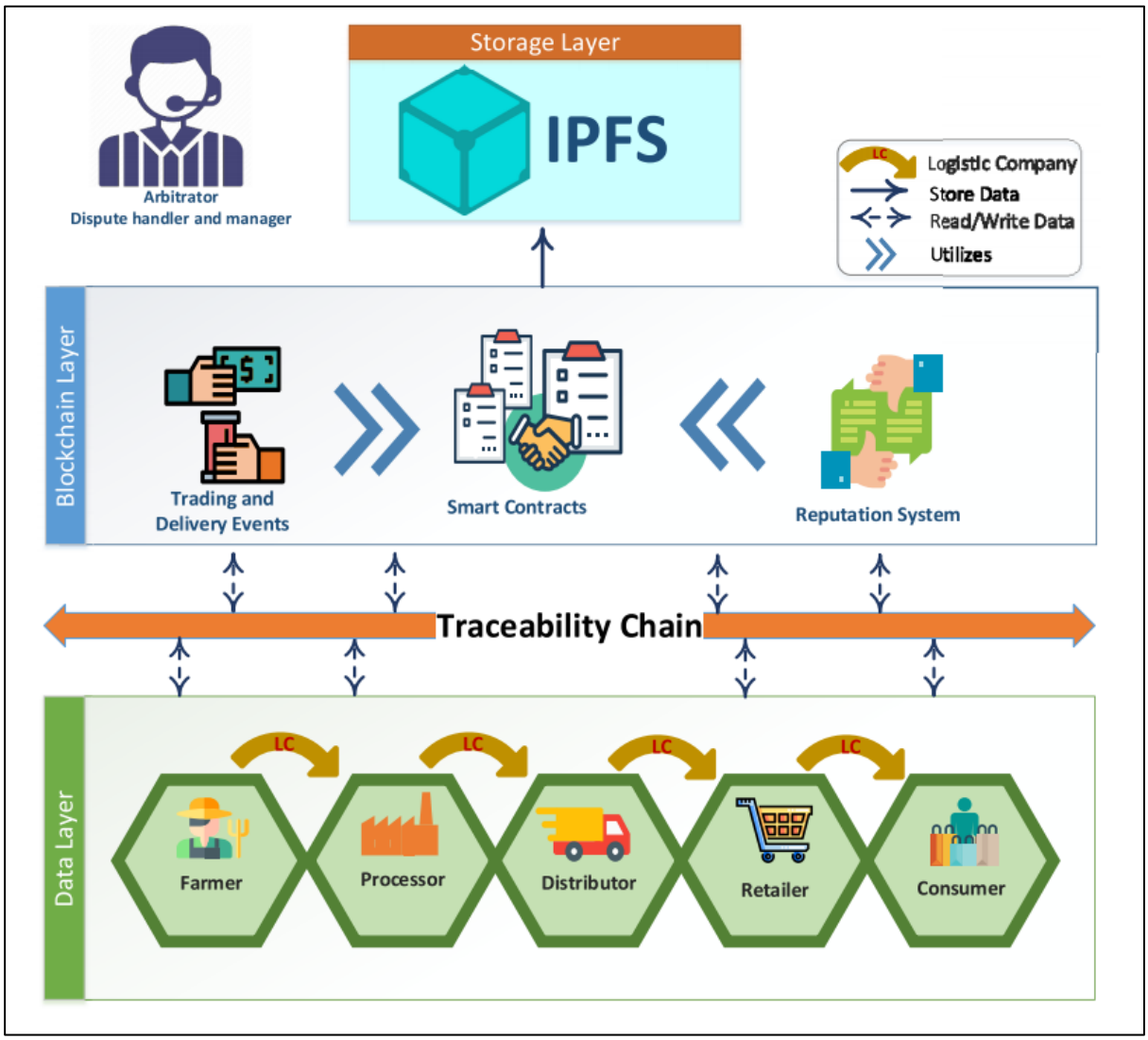


Figure 2.2: Blockchain-based end to end solution for agri-food supply (Shahid et al., 2020).

2.4 A Blockchain-Driven Food Supply Chain Management Using QR Code and XAI-Faster RCNN Architecture

This paper focuses on the food supply chain management system based on blockchain technology that integrates QR codes, Explainable Artificial Intelligence(XAI), and Region Convolutional Neural Network (RCNN) for food traceability and verification. The paper

presents a unique solution for identifying and verifying food products by employing a QR code and simultaneously using XAI technology that enhances transparency and comprehensibility of the system's operation. Furthermore, the paper tests and assesses how well the proposed system performs with different efficiency ratios on food data to confirm its effectiveness for safety, traceability, and verification of food data. The paper concludes that it is better than the other existing methods in terms of speed and accuracy of results, achieving 99.53% accuracy.

The suggested system has several advantages over the traditional system. First, the use of blockchain alongside the integration of QR codes allows for the preservation of food traceability and transparency in a highly efficient manner. Users can easily get information regarding the food's production, processing, and storage, meaning there is a guarantee of food safety and enhanced user trust. Secondly, the system implements XAI technology, which makes the operating methods of artificial intelligence more understandable for the users. This helps the users believe in the output of the machine learning algorithms. Moreover, the Faster RCNN architecture adequately detects and monitors any suspicious harmful content of the food, thus guaranteeing the food's health safety. The experimental findings of the system show that it responds faster, has greater accuracy in processing, and is easily adaptable and pragmatic, which outclasses already existing methods.

This paper has a few shortcomings. First, the paper does not consider the plausibility of smart contracts of tokenization mechanisms, which are usually system components, as they are developed to execute contract terms or incentives on the blockchain automatically. Second, while the system claims to prioritize food traceability and information security, little attention is given to the methodology of implementing these issues in real-world situations, particularly the system's scalability, interoperability, and lack of thorough operational verification. Furthermore, the article does not discuss the latency challenge involved in blockchain

computing and the stable application in real settings of traceability devices like sensors, forming a potential limitation to the system's practical application.

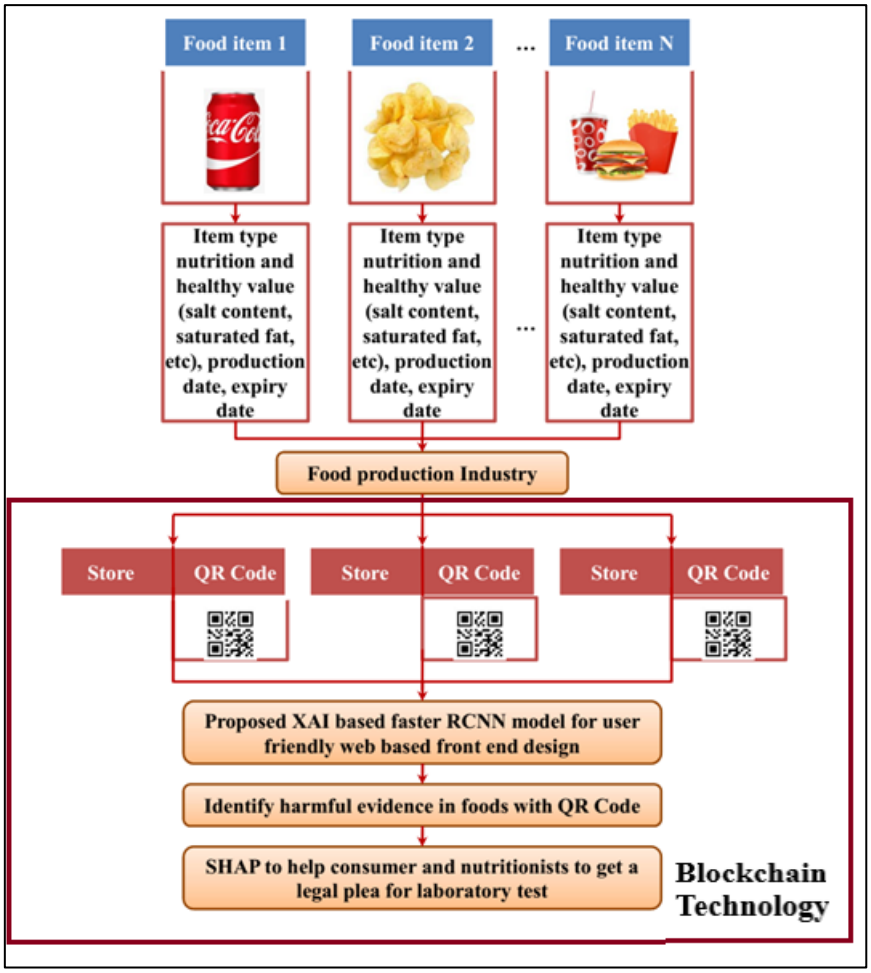


Figure 2.3: Overall Architecture of Blockchain-driven XAI-based Faster RCNN Model (Bhatia & Albarrak, 2023).

2.5 Comparison between Papers

Table 2.1: Comparison of Blockchain-based Food Supply Chain Management Papers.

	Paper 1 (Blockchain-based Traceability in Agri-Food Supply Chain Management: A Practical Implementation)	Paper 2 (Blockchain-based Agri-food Supply Chain: A Complete Solution)	Paper 3 (A Blockchain-Driven Food Supply Chain Management Using QR Code and XAI-Faster RCNN Architecture)
Transparency	✓	✓	✓
Traceability	✓	✓	✓
Scalability	✓	✓	✓
Tokenization	✗	✗	✗
Smart Contract	✓	✓	✗
QR Code Integration	✓	✗	✓
IPFS Integration	✗	✓	✗
Automated Refund	✗	✗	✗

All three papers focus on transparency as a critical component in their studies. Paper 1 tracks product information throughout the chain and understands blockchain technology as the foremost solution for transparency toward stakeholders in food supply chains. Of equal significance, Paper 2 portrays the use of blockchain in agricultural supply chains as fundamental

in supporting the transparency of food products. Paper 3 builds a transparent system where consumers trace the origin and the quality of food by integrating blockchain with QR codes.

All the papers incorporate blockchain to trace food items back to their source. Paper 1 refers to graph technology to enhance food safety and quality assurance through end-to-end traceability. Paper 2 uses IoT applications with blockchain technology, improving food traceability from farm to plate. Paper 3 combines QR codes with blockchain for enhanced evidence-traceability and secure access to product information.

The considerations regarding scalability are handled to different extents. Paper 1 does not mention scalability issues, yet blockchain's use in extensive supply chains gives the impression that it may accommodate more data. Paper 2 mentions scalability issues regarding the blockchain's capacity for real-time data and proposes a solution using IPFS storage. Paper 3 demonstrates scalability by evaluating the system's performance under different payload sizes and user loads, proving that the system can handle large-scale data effectively.

In terms of tokenization, none of the three papers explicitly discuss this aspect. While blockchain is central to each paper's methodology, they do not delve into tokenization as a specific mechanism within their proposed systems.

Only Papers 1 and 3 incorporate QR codes to improve accessibility and openness. Paper 1 integrates QR codes to relay relevant traceable information to consumers regarding food products. Paper 2 does not mention QR codes as part of its system. Instead, it focuses exclusively on using blockchain and smart contracts to track and trace food products within the supply chain. Paper 3 places great attention on QR codes as the primary method for consumers to obtain digitized food product information, including verification of product details and traceable information.

Regarding IPFS integration, Paper 2 is the only paper that explicitly integrates **IPFS** as a storage solution. It uses IPFS to store data off-chain, while blockchain handles the transaction and validation of data. Papers 1 and 3 do not mention using IPFS for storage purposes.

None of the three papers discussed the integration of IPFS, and instead, they seem to have concentrated their efforts on blockchain and QR code technology for tracing and transparency purposes. Lastly, none of the papers mention automated refund mechanisms, emphasizing improving tracing, transparency, and information security in the data rather than customer service processes like refunds.

2.6 Blockchain

Blockchain is a form of distributed ledger technology that keeps a record of transactions or events. The information is stored in encrypted blocks to ensure privacy. The blocks are arranged sequentially and form a chain in which each block contains several transaction records. Each block is linked to each other using a cryptographic hash and made the cryptographic security characteristics of the blockchain (Iftekhar et al., 2020). Consensus algorithms like Proof of Work(PoW) and Proof of Stake (PoS) are responsible for validating and securing the transaction across the network and additionally improving cryptographic security. (Bodkhe et al., 2020; Tian, 2016).

Blockchain is also an innovative type of transparent, decentralised database for transaction data. Every network computer, referred to as a node, works together to operate the database. The ledger would be accessible to every node involved in the transaction through various devices (Javaid et al., 2021). This feature guards against unauthorised tampering and provides an unalterable transparent record of each transaction. It enables real-time data sharing among all stakeholders, enhancing transparency and confidence in a variety of applications,

including supply chain management and financial transactions (Galvez et al., 2018; Kshetri, 2017).

Blockchain technology is defined by several features that differentiate it as a game-changing technology in many fields. Blockchain can be defined as a distributed ledger system that is decentralized in its architecture and transparent due to the cryptographic locks on the various block data. As each block holds data, it is linked to other blocks by cryptographic pointers or hashes, which creates a secure chain. This system supports the claim for the verification of data integrity since any changes to a block will compromise the links, leading to an alert for potential tampering to all users who are connected to the system (Bodkhe et al., 2020). Furthermore, the absence of a central authority makes these systems more secure since there are no possibilities of central control that can threaten blockchains, enabling the recording and sharing of transactions over a network with greater ease (Bodkhe et al., 2020).

Moreover, Blockchain maintains the ability of providing traceability and transparency, which is among the core features that are required within supply chains and other industries that rely on the provenance of goods (Galvez et al., 2018). The elimination of a central authority of control signals another advantage as secure real time data sharing becomes possible without risks, helping to establish confidence in participants within the network (Javaid et al., 2021).

One key advancement of blockchain is the smart contracts that enable the automation of processes when specific conditions are met and are stored directly onto the blockchain as self-executing agreements. Such contracts lessen dependency on intermediates which improves effectiveness while lowering costs in finance, insurance, logistics, among others towering (Piccardo et al., 2024). Besides, blockchain's efficiency stems from its capacity to increase productivity by removing manual work as well as unused intermediaries, which drives down expenses and increases the speed of transactions (Ellahi et al., 2024; Tripathi et al., 2023).

To finalize, blockchain interoperability is emerging as a crucial feature for the communication between distinct platforms and systems. This helps as the adoption of blockchain steadily rises across sectors enabling different networks to come together and interface successfully (Javaid et al., 2021). In sum, blockchain is indeed an incredible technology that can change the world for the better. With its features like decentralization, immutability, transparency, security, and traceability, it has got the potential to overcome set barriers with proper systems in place.

2.7 History of Blockchain

Blockchain technology has undergone significant transformation owing to several notable past inventions. In 1979, David Chaum proposed a distributed computing system that supported secure interaction and interactivity with users in a hostile environment (Fiore & Mongiello, 2023). In 1983, Chaum also coined the concept of blind signatures, a form of cryptographic digital payment that offered anonymity to its users and marked the beginning of such privacy-focused technologies (Tripathi et al., 2023). Progress was continued in 1991 by Stuart Haber and W. Scott Stornetta, who proposed using timestamps to create a chain of blocks secured by encryption to protect the integrity of digital documents (Bodkhe et al., 2020; Fiore & Mongiello, 2023). Their research formed the basis for creating the crucial aspect of blockchain, the immutable ledger. Hal Finney proposed a Reusable Proof of Work (RPoW) 2004 (Tripathi et al., 2023).

The genesis of blockchain technology is referred to as blockchain 1.0, and it started in 2008 with Satoshi Nakamoto's use of Bitcoin. During this time, the purpose of incorporating blockchain as a decentralized technology was enabled into a platform that could facilitate user-to-user financial transactions without any intermediaries. The epoch-making invention of

Blockchain 1.0 is that it was for the first time possible to use cryptography and consensus-based approaches such as Proof of Work (PoW) to record and validate transactions safely. This provided a framework of robust transparency and immutable notwithstanding the many changes which would occur over time. The pinnacle of Blockchain 1.0 was widely recognized to be its role in the making and management of bitcoin with the latter serving as an alternative to currency substitutes (Bodkhe et al., 2020; Fiore & Mongiello, 2023; Tripathi et al., 2023).

The introduction and use of smart contracts made programmable functionalities possible on the blockchain, and so here comes the second generation, Blockchain 2.0. This generation enabled other blockchain applications besides simple cryptocurrency transactions and provides services in the areas of healthcare, government, science. A significant development in this generation was the creation of the Ethereum platform by Vitalik Buterin in 2015 to deploy and execute smart contracts. Other platforms like Hyperledger also entered the market, opening the door for enterprise-level blockchain applications in various industry (Bodkhe et al., 2020; Tripathi et al., 2023).

As blockchain technology matures into the third generation, attempts have been made to address scalability, interoperability and sustainability issues. In this phase, the convergence towards the decentralized applications was introduced. Blockchain was developed beyond finance to cover energy, manufacturing, education and governance (Bodkhe et al., 2020; Tripathi et al., 2023).

Blockchain 4.0 leverages public distributed databases for real-time synchronization of services and enables integration with Industry 4.0 for smarter and more efficient automation and connectivity. It uses smart contracts to replace paper-based agreements and pave the way for automated and consensus-based operations on the network (Bodkhe et al., 2020). This generation includes artificial intelligence (AI) and digital intelligence, resulting in improved

systems that can predict, adapt and automate without central control, further establishing blockchain as an important part of innovative and changing ecosystems across the programming world (Tripathi et al., 2023).

CHAPTER 3 REQUIREMENT ANALYSIS AND DESIGN

3.1 Introduction

This chapter explains in detail how the proposed blockchain application would be developed. Each phase of software development is explained in detail. A software development methodology is a framework used to structure, plan, and manage the software development process (Nikitin, 2024). Ideal methods are always a compromise in terms of their comparative advantages and disadvantages. Therefore, it is imperative to have a method that best suits the needs of the software development process (McGuire, 2024). Since it specifies which activities should be carried out and which processes should be applied, the methodology chosen here improves the control and monitoring of the development process. In this proposed application, the author decided to use the Agile Software Development Life Cycle as a guide for developing the application.

3.2 Software Development Life Cycle

3.2.1 Requirement Analysis

Requirement analysis involves anticipation and integration of the wishes of stakeholders in every phase of a project, preparation for the new system, and analysis in the software development life cycle (SDLC). In this phase, the utmost significance is set on defining the purpose of the final product to ensure its usability and business impact. In this stage, the description of functional and non-functional requirements is elaborated, constraints and limitations are addressed, and business value derives different criteria prioritization. Addressing the requirements of all stakeholders and subsequent reviews is critical to mitigate risks that stem from misunderstanding the goals.

3.2.1.1 Requirement Elicitation

The elicitation of system requirements is a critical step in ensuring the successful design and development of the proposed blockchain-based food supply chain system. In this section, recent scholarly works are used to identify and validate functional and non-functional needs. In this manner, the system requirements are developed by analysing existing studies to address industry challenges and user needs.

A systematic literature review was done to gain understanding of how blockchain technology is utilized in food supply chain systems. Key databases like, IEEE Xplore and ScienceDirect were accessed. The selection criteria included:

- Application of blockchain in supply chain management.
- Relevance food traceability, ownership of products, logistics and utilization by users.
- Published studies between 2017 and 2024.

The review defined best practices and related functions that were elicited into system requirements for this project.

Author	Justification	Related Functions
Ellahi et al., 2024	It is mentioned that blockchain technology is used to manage role permissions in the food supply chain, generate unique Token IDs, and ensure traceability and ownership records of the product life cycle.	• User Roles and Permissions • Product Management • Ownership Management • Logistics Data Management
Ahmad & Bailey, 2021	The blockchain-based product tracking function is described, including full life	• Product Management

	cycle tracking from production to consumption, and the scenario in which consumers verify product authenticity through QR codes is discussed.	• Consumer Access and Verification • Data Integrity and Traceability
Piccardo et al., 2024	It discussed how to realize user role verification and permission allocation through smart contracts, simplifying public service management, and mentioned the automatic refund mechanism based on blockchain.	• Role-Based User Registration • Product Refund Management
Kshetri, 2017	Emphasis on the role of blockchain in logistics data management, ownership transfer and supply chain transparency, especially its contribution to reducing fraud and improving efficiency	• Ownership Management • Logistics Data Management • Data Integrity and Traceability
George et al., 2024	It discusses how consumers can access product information stored on the blockchain through QR codes to verify product authenticity and view complete supply chain information.	• Consumer Access and Verification

3.2.1.2 Functional Requirements

Functional requirements specify system behaviours or functions, such as the activities that a particular system should perform. They are basic requirements for the successful fulfilment of the purpose of any system and usually include the functions that deal with data processing, user criteria checking and user-system interactions. They provide clear answers to the "what" questions regarding a system's performance and serve as a foundation for system developers and other interested parties.

1. User Roles and Permissions
    - The system should support the following roles:
        i. Farmer: Can add and manage products and batches.
        ii. Logistic: Can update logistics data and transfer ownership.
        iii. Retailer: Can receive products from logistics and transfer ownership to consumers.
        iv. Consumer: Can scan QR codes, view product details, and verify authenticity.
    - Each role should have predefined permissions managed through a smart contract.
    - The smart contract must enforce that users can only perform actions allowed by their role.
2. Role-Based User Registration
    - Users should be able to self-register through the system.
    - During registration, users must select a role (Farmer, Logistic, Retailer, Consumer).
    - The smart contract should validate the user's registration and assign the selected role.
    - New user registrations should be stored on-chain and accessible to all participants.
3. Role Verification
    - The system should verify a user's role before allowing any action.

- The smart contract must ensure:
    i. Only Farmers can add products and batches.
    ii. Only Logistics users can update transportation details.
    iii. Only Retailers can accept product deliveries.
    iv. Consumers can only view and verify product data.

4. Product Management by Farmer
    - Farmers should be able to:
        i. Add new products, including metadata like name, batch number, production date, and location.
        ii. Generate unique Token IDs for each product upon creation.
        iii. Generate QR codes linked to the Token ID or IPNS address.
        iv. View all products and batches they have created.
        v. Track the status and history of each product.
5. Ownership Management
    - The smart contract should track the ownership of each product token.
    - Ownership transfer should be possible only through a valid transaction:
        i. Farmer → Logistic: Upon product dispatch.
        ii. Logistic → Retailer: Upon delivery to a retailer.
        iii. Retailer → Consumer: Upon purchase by a consumer.
    - Ownership history must be stored on-chain and immutable
6. Logistics Data Management
    - Logistic users should be able to:
        i. View assigned products for transportation.
        ii. Update logistics data, such as start date, arrival date, and timestamp.

iii. Upload updated metadata to IPFS and link it to the product's Token ID.

iv. Transfer ownership to the next party upon delivery.

7. Consumer Access and Verification

- Consumers should be able to:

i. Scan QR codes to access product data.

ii. View product details, including production, logistics, and ownership history.

iii. Receive the warning if any mismatch of product data is detected.

iv. View the token information linked to the product they purchased

8. Product Refund Management

- Consumers should be able to select the product to refund.
- The refund process should be execute automatically.

9. Blockchain and IPFS Integration

- All product metadata should be uploaded to IPFS and linked to the Token ID.
- The system should validate IPFS data by comparing the hash with the blockchain record.
- IPFS Content ID and ownership transactions should be stored immutably on the blockchain.

10. Mobile and QR Code Support

- The system should allow users to scan QR codes using mobile devices to:

i. Access product data.

ii. Verify product authenticity on the blockchain.

11. Token Management

- The system should automatically create a blockchain-based token for each product.
- The token must:

i. Represent the product uniquely

ii. Be-linked to the product's IPFS metadata

iii. Track ownership and transfer history

12. Decentralized Access Control

- The smart contract should implement decentralized access control:
    i. Users cannot modify or access data outside their permissions.
    ii. Unauthorized actions must be rejected by the smart contract.

13. Data Integrity and Traceability

- The system should store an immutable history of all actions on the blockchain, including:
    i. Product creation and updates.
    ii. Logistics data updates.
    iii. Ownership transfers.
- Consumers and other users should be able to trace the full lifecycle of a product back to its origin.

3.2.1.3 Non-Functional Requirements

Non-functional requirements address the ‘how’ of a system with an emphasis on performance metrics such as response time or system scalability, as well as usability, security, and maintainability standards. These requirements guarantee that a product is operational, reliable, and secure. Non-functional requirements aid in the performance attributes and limits of the system, making sure everything is done perfectly. These requirements make it possible for the system to operate seamlessly and as the functionals expect. These have a tangible aspect to them aiding in the way the system is focused at and aiding in the user experience.

1. Performance
    - The average response time for retrieving product data via QR code should not exceed 3 seconds.
    - Blockchain transactions should be completed within 15 seconds, depending on the network congestion.
2. Security
    - The system should implement role-based access control to restrict actions to authorized users.
    - Unauthorized or suspicious actions must trigger alerts.
3. Usability
    - Mobile-friendly design to support users accessing the system on smartphones.
    - QR code scanning must be seamless, and the redirected pages should be easy to navigate.
4. Maintainability
    - The system must be modular, allowing for updates to specific modules easily.
    - Smart contracts should be upgradeable to accommodate future feature additions or bug fixes.
5. Transparency and Auditability
    - All blockchain transactions must be publicly accessible via blockchain explorers.
    - The system should maintain comprehensive logs of all actions performed by users, stored immutably on the blockchain for auditing purposes.

3.2.2 System Design

3.2.2.1 Architecture Design

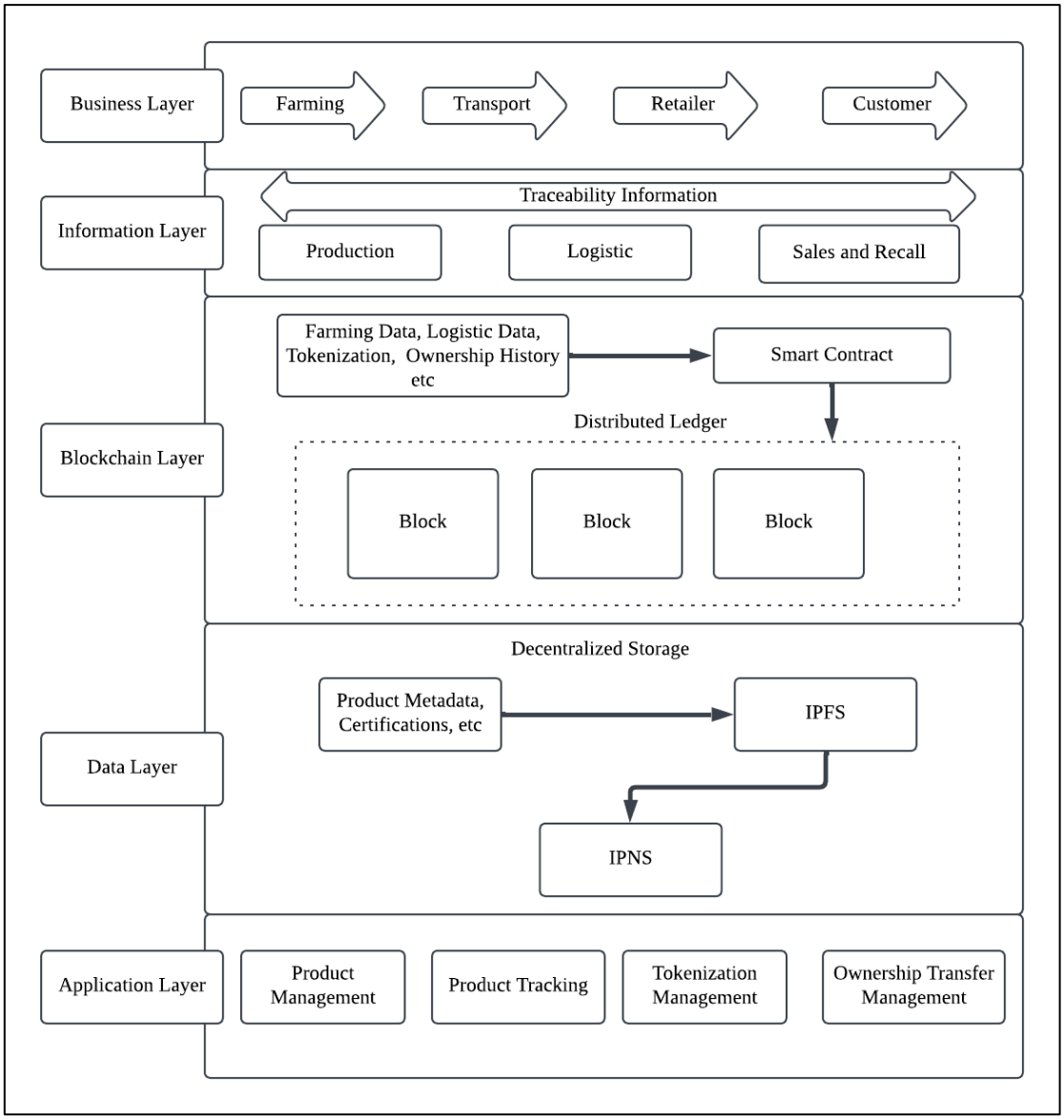


Figure 3.1: System architecture for blockchain-based food supply chain system

Figure 3.1 shows the system architecture represents a decentralized blockchain-based food supply chain management platform. The design is divided into multiple layers: Business Layer, Information Layer, Blockchain Layer, Data Layer, and Application Layer. Each layer has distinct responsibilities, working together to ensure transparency, traceability, and integrity in the supply chain. Key components include smart contracts for blockchain operations, IPFS for

decentralized data storage, and application modules for product management and tracking. The architecture enables seamless collaboration between stakeholders like farmers, logistics providers, retailers, and consumers while maintaining data security and decentralization.

3.2.2.2 Context Diagram

A context diagram provides a high-level overview of a system, illustrating how external entities interact with the proposed system. It defines the main data inputs into the system and the outputs generated by the system for each actor. Figure 3.2 shows that the context diagram of the proposed system and the external entities. The external entities will be the Farmer, Logistic, Retailer, and Customer. The Farmer provides input in the form of product data and receives output in the form of a token ID and product status, which confirm the tokenization and tracking of the product. The Logistic actor inputs logistics data and receives delivery confirmation as an output, acknowledging the successful transfer of ownership or updated status. The Retailer submits product receive information to confirm the acceptance of goods and retrieves product information and history. Finally, the Consumer interacts with the system by scanning a QR code to access product details and ownership history, which are verified for authenticity through blockchain records.

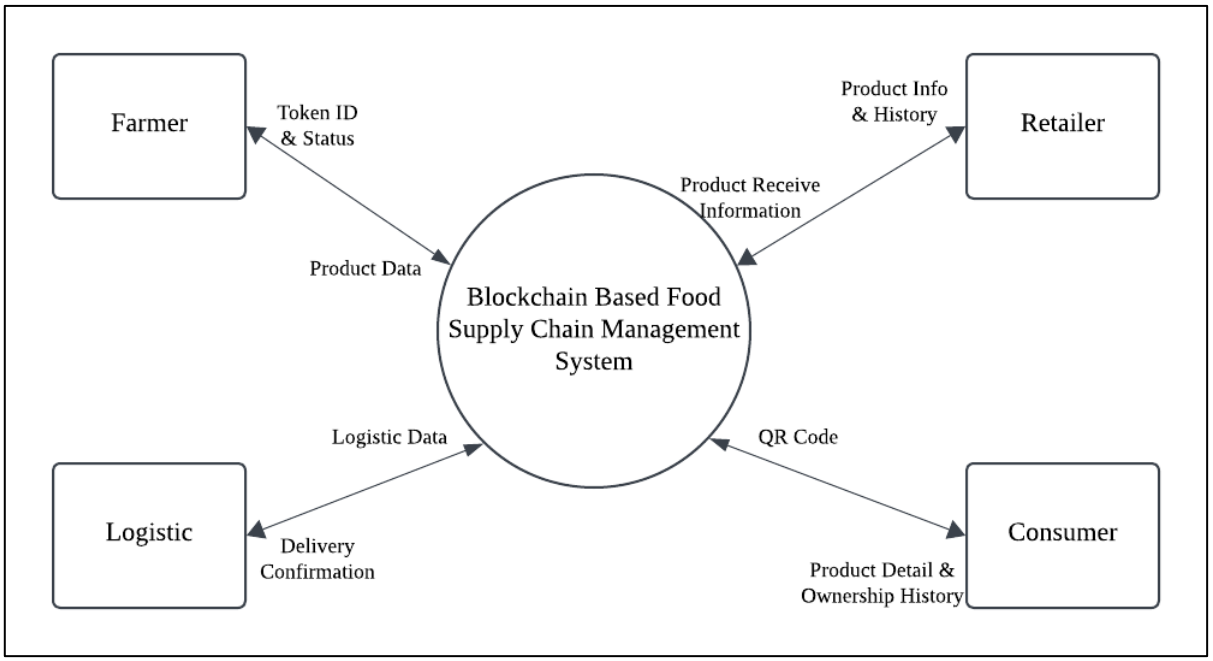


Figure 3.2: Context diagram between proposed system and external entities.

3.2.2.3 Data Flow Diagram

A data flow diagram (DFD) exhibits how information within a system is processed and moves between processes, data collections, and external entities. The DFD's primary goal is to portray the system's general features in a way that makes it easy to study, formulate, and elaborate on its functional needs.

3.2.2.3.1 Level 0 Data Flow Diagram

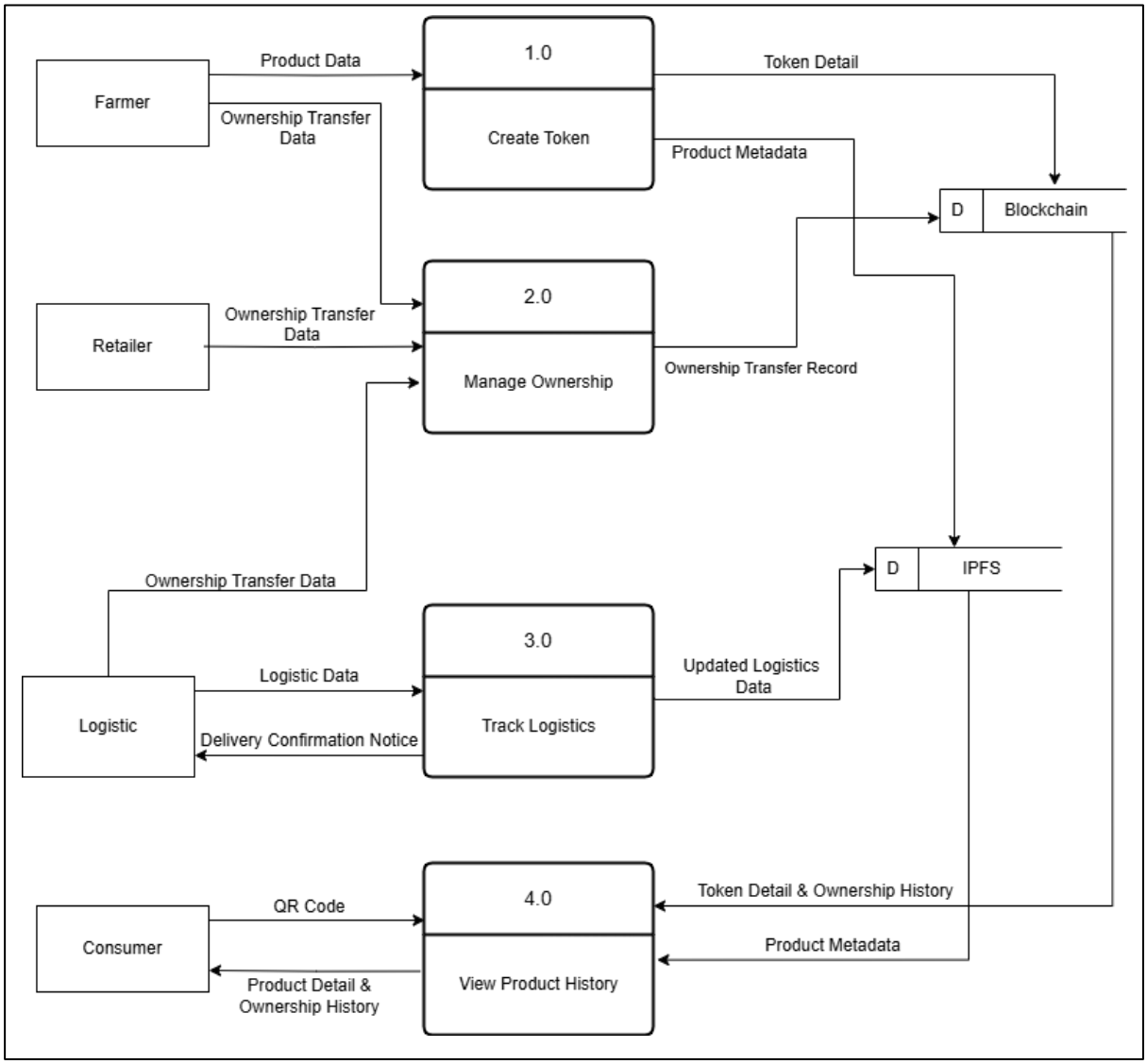


Figure 3.3: Level 0 DFD of proposed system.

Figure 3.3 shows the Level 0 Data Flow Diagram (DFD) which provides a high-level overview of the proposed blockchain-based food supply chain system. It illustrates the interaction of data between the primary actors (Farmer, Logistic, Retailer, and Consumer) and the system. The system is divided into several main data processes such as create token, manage ownership, track logistic, and view product history. Data is stored securely on Blockchain and IPFS to ensure transparency, immutability, and traceability. The process begins with Process

1.0: Create Token, where the farmer inputs product data, which is stored on IPFS, and a unique token is generated on the blockchain. In Process 2.0: Manage Ownership, the farmer, logistic provider, and retailer update ownership transfer data, ensuring a tamper-proof history recorded on the blockchain. Process 3.0: Track Logistics involves the logistic provider updating transport conditions, such as location, which are stored on IPFS and validated on the blockchain. Finally, in Process 4.0: View Product History, consumers can scan a QR code to access product details stored on IPFS and verified via the blockchain.

3.2.2.3.2 Level 1 Data Flow Diagram

The expanded form of Level 0 DFD is provided by a Level 1 DFD by completely decomposing the main processes of the circumscribed system into several sub processes which provides a deeper understanding of the working of the system. It shows how data and information are transferred between sub processes, external entities, and data stores to give a better explanation of what goes on inside the working of the system.

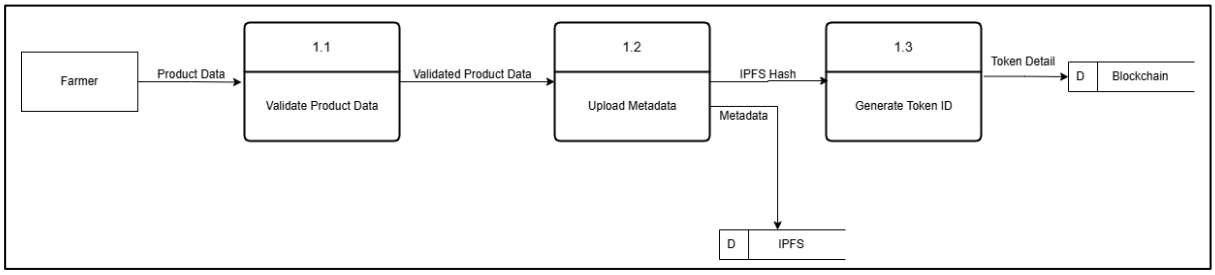


Figure 3.4: Level 1 DFD for Process 1.0 Create Token.

Figure 3.4 shows the Level 1 DFD which details Process 1.0 Create Token demonstrates the generation of a unique blockchain based token ID from product data provided by the farmer. It comprises of an IPFS hash, a token ID, and blockchain data. The process starts with checking the input information and then proceeds with getting the metadata uploaded on IPFS and

creating a Token ID. The final outputs are the token ID and the IPFS hash stored in the blockchain and IPFS storage.

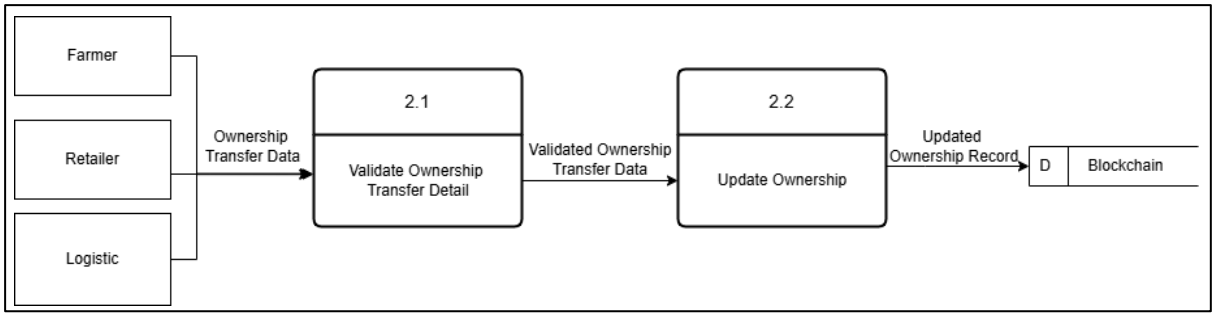


Figure 3.5: Level 1 DFD for Process 2.0 Manage Ownership.

Figure 3.5 illustrates Level 1 DFD regarding Process 2.0: Manage Ownership provides a clear overview of the entire process of product ownership transfer between actors like Farmer, Logistic, or Retailer. Ownership transfer data are validated before being updated in the system. After the product ownership record is updated, it is uploaded and saved in the Blockchain whereby enabling traceability and transparency.

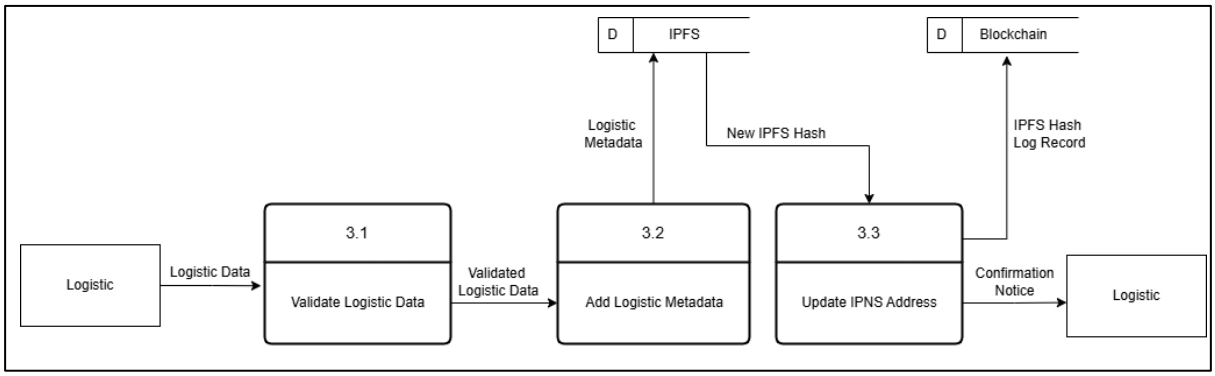


Figure 3.6: Level 1 DFD for Process 3.0 Track Logistics.

The Level 1 DFD of Process 3.0: Track Logistics shows as Figure 3.6. It outlines how logistics data from the Logistic entity is entered, handled and recorded within the system. This process includes checking logistics data validity, taking the metadata and storing in IPFS as well as updating the IPNS record so that the latest logistics metadata is accessible. Also, the IPFS hash data is etched into the Blockchain, and the system send a confirmation notice to Logistic entity.

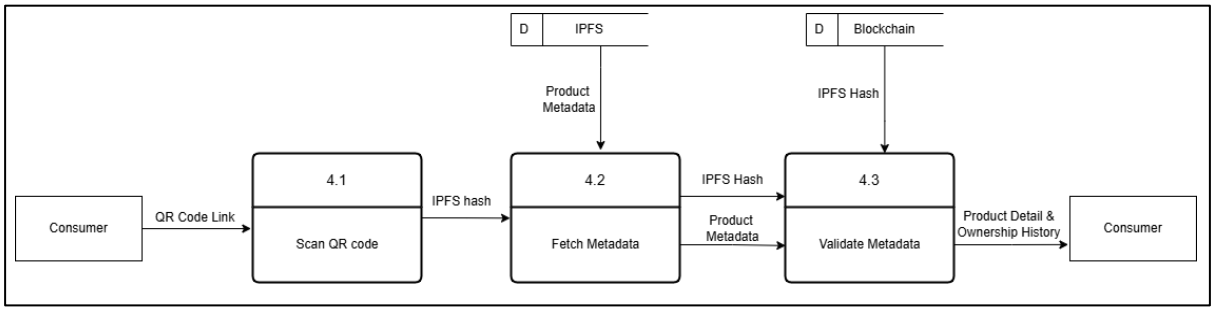


Figure 3.7: Level 1 DFD for Process 4.0 View Product History.

Figure 3.7 displays Level 1DFD for Process 4.0: View Product History. It illustrates how a Consumer retrieves the product details by scanning a QR code. The process involves fetching metadata from IPFS, validating the data's integrity against the Blockchain, and displaying the product's verified details, including its ownership history and production traceability, to the Consumer.

3.2.2.4 Use Case Diagram

A Use Case Diagram graphically describes the goals of the users or external systems and their interactions with the system. It illustrates the requirements of the system by showing use cases, actors and their associations. Use case diagrams portray the functions of the system briefly, which is useful for stakeholders to grasp the system user interaction and its limits. The use case diagram for the proposed system is shown in Figure 3.8, which highlights the main use cases and how relevant actors are involved with them.

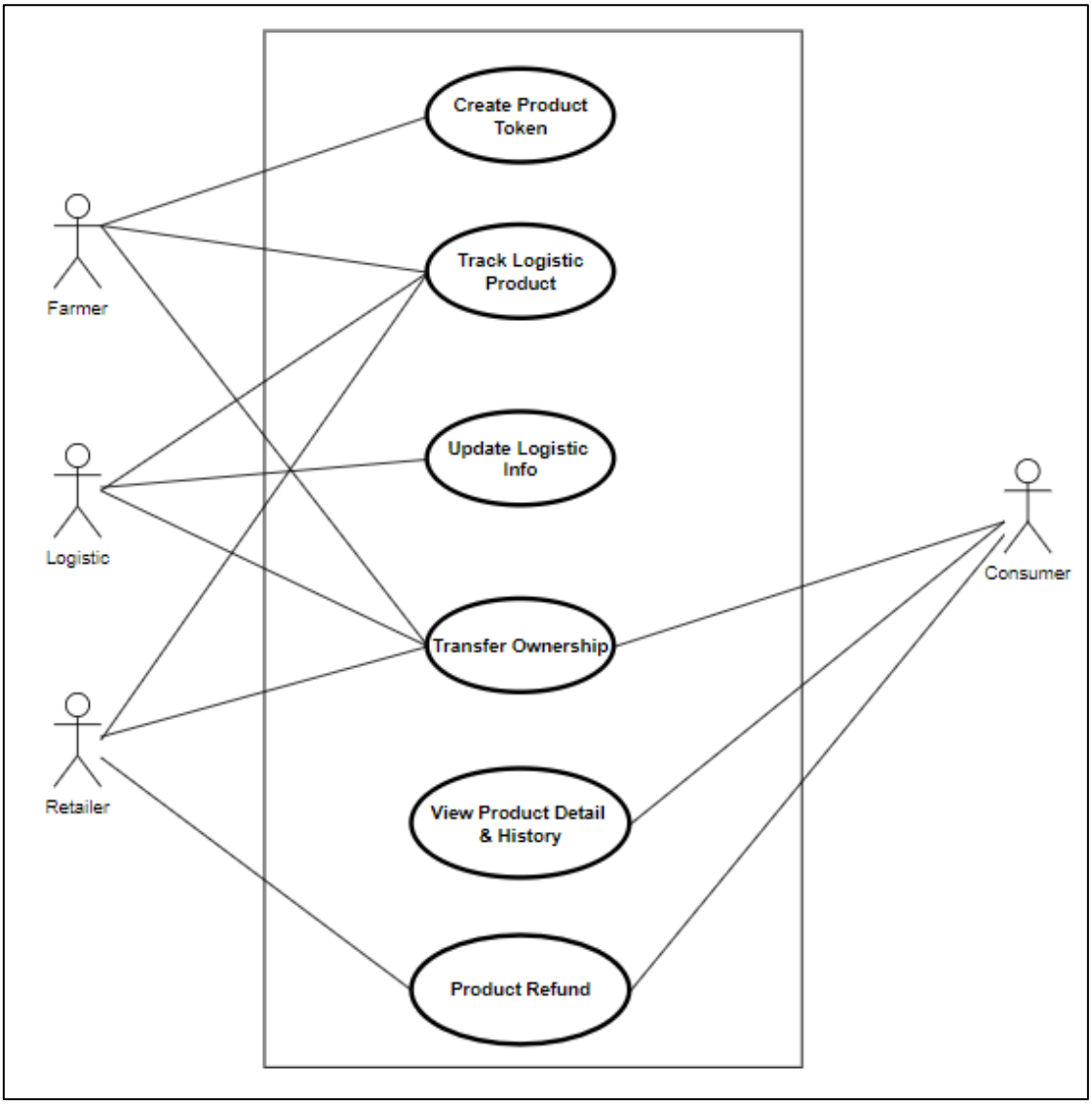


Figure 3.8: Use Case Diagram for proposed system.

3.2.2.5 Use Case Specification

Use Case Specification outlines a particular use by an actor for the system to accomplish a process. It elaborates the functional needs of the system in a prescribed way. These being the description, preconditions, postconditions, major flows, and other specifics for all use cases. The Use Case Specifications for each identified use case in the system are detailed as follows.

Table 3.1: Use Case Specification for "Create Product Token".

Use Case Name	**Create Product Token**
Actor(s)	Farmer
Description	Create a unique blockchain token for a product and store related metadata securely.
Preconditions	Farmer must be authenticated and logged in.
Postconditions	• A blockchain token is created and associated to the product metadata. • Metadata is stored in IPFS, and the respective IPFS hash is recorded on the blockchain. • A QR code containing the tracking URL is generated.
Main Flow	1. Farmer logs in system and selects “Add New Product”. 2. Farmer inputs product details such as name, batch number, and production date etc. 3. System validates the input data. 4. System uploads validated product metadata to IPFS and getting respective IPFS hash 5. System creates a unique Token ID and stores it with the metadata IPFS hash on the blockchain.

	6. System generates a QR code containing the tracking URL and token ID for each product. 7. The system confirms that the token, and QR code generation were successful. 8. System displaying the Token ID, IPFS hash, and QR code to the Farmer.
Alternate Flow	1. Invalid Product Data: i. System detects incomplete or invalid data. ii. Farmer is notified to correct the input and retry. 2. IPFS Upload Failure: i. The system logs the issue and notifies the Farmer 3. Blockchain Token Creation Failure: i. The system logs the issue and notifies the Farmer 4. QR Code Generation Failure: i. The system logs the issue and notifies the Farmer

Table 3.2: Use Case Specification for "Track Logistic Product".

Use Case Name	**Track Logistic Product**
Actor(s)	Farmer, Logistic, Retailer
Description	Actors able to track the status and location of products during their logistics process.
Preconditions	• The Farmer or Logistic or Retailer actor must be authenticated and logged into the system. • The product must have been handed off to the Logistic actor with proper ownership transfer recorded on the blockchain.
Postconditions	The Farmer or Logistic or Retailer actor successfully views: • The status and location of the product. • A complete history of logistics updates, including timestamps. • Confirmation of whether the product has reached the Retailer.
Main Flow	1. The Farmer or Logistic or Retailer actor logs into the system and navigates to the “Logistic Tracking” page. 2. The user selects a specific product by apply filter such as token ID, batch number, etc. 3. The system retrieves the latest IPFS hash from the blockchain. 4. The system retrieves the latest logistic data from IPFS based on IPFS hash. 5. The system displays the details about location, transport status, timestamps, and delivery confirmation.

Alternate Flow	1. Blockchain Retrieval Failure: i. The system logs the issue and notifies the user. 2. IPFS Retrieval Failure: i. The system logs the issue and notifies the user.

Table 3.3:Use Case Specification for "Update Logistic Data".

Use Case Name	**Update Logistic Data**
Actor(s)	Logistic
Description	Update the product's logistics information
Preconditions	• The Logistic actor must be authenticated and logged into the system. • The product must have a valid Token ID stored on the blockchain.
Postconditions	• The new logistics metadata is successfully uploaded to IPFS, generating a new IPFS hash. • The blockchain is updated with the new IPFS hash. • The IPNS address is updated to point to the new IPFS hash.
Main Flow	1. The Logistic actor logs into the system and navigates to the "Update Logistic Data" page. 2. The Logistic actor selects the specific product to update by apply filter such as token ID, batch number, etc. 3. The Logistic actor inputs the updated logistics data. 4. The system validates the logistics data. 5. The system generates new metadata by appending the updated logistics data to the existing metadata. 6. The system uploads the updated metadata to IPFS, generating a new IPFS hash. 7. The system updates the IPNS address to point to the new IPFS hash. 8. The system writes the new IPFS hash to the blockchain, linking it to the product's Token ID.

	9. The system confirms the successful update to the Logistic actor.
Alternate Flow	1. Invalid Logistics Data: i. The Logistic actor is notified with a message indicating the specific errors. ii. The system prompts the actor to correct the data and retry. 2. IPFS Upload Failure: i. The system logs the issue and notifies the Logistic actor. 3. Blockchain Record New IPFS Hash Failure: i. The system logs the issue and notifies the Logistic actor.

Table 3.4: Use Case Specification for "Transfer Ownership".

Use Case Name	**Transfer Ownership**
Actor(s)	• Current Owner: The entity transferring the product ownership (Farmer, Logistic, Retailer). • New Owner: The entity receiving the product ownership (Logistic, Retailer, Consumer).
Description	Transfer ownership of a product to the next entity in the supply chain
Preconditions	• The current owner must be authenticated and logged into the system. • The product must have a valid Token ID and associated metadata stored on the blockchain. • The new owner must have an active account in the system.
Postconditions	The product's ownership is successfully transferred to the new owner. • The blockchain is updated with the ownership transfer record. • The product's ownership history is retrievable for traceability.
Main Flow	1. The current owner logs into the system and navigates to the "Transfer Ownership" page. 2. The current owner selects the product(s) to transfer. 3. The current owner enters the new owner's account ID. 4. The system verifies the existence of the new owner in the system. 5. The system prompts the current owner to confirm the transfer. 6. The system generates a transfer request and records the initiation timestamp. 7. The system notifies the new owner of the pending transfer request. 8. The new owner reviews the pending transfer request. 9. The new owner chooses to approve the request. 10. The system records the ownership transfer on the blockchain includes token ID, previous owner's account ID, new owner's account ID, and timestamp. 11. The system updates the product's metadata to include the new owner in the ownership history.

	12. The system notifies both the current owner and the new owner of the successful transfer.
Alternate Flow	1. New Owner Not Found: i. The current owner is notified with a message like “New owner not found. Please verify the account details.” ii. The process is terminated unless a valid new owner is provided. 2. Transfer Confirmation Timeout: i. The current owner does not confirm the transfer within a specified timeframe. ii. The system cancels the operation and notifies the current owner. 3. New Owner Reject Request: i. The system updates the request status to “Rejected.” ii. The system notifies the current owner of the rejection. iii. The process terminates. 4. Blockchain Write Failure: i. The system logs the error and notifies the actor of the issue.

Table 3.5: Use Case Specification for "View Product Detail and History".

Use Case Name	**View Product Detail and History**
Actor(s)	Consumer
Description	Access detailed product information, including its production, logistics, and ownership history, by scanning a QR code.
Preconditions	• The consumer has a mobile device or system capable of scanning QR codes. • The product's metadata is stored in IPFS, with the corresponding IPNS address recorded in the QR code. • The blockchain contains the product's Token ID, associated IPFS Hash and IPNS Address for validation.
Postconditions	• The consumer successfully views the product's detailed information and history.
Main Flow	1. The consumer scans the QR code on the product packaging using a mobile device or system. 2. The system extracts the IPNS address from the QR code. 3. The system retrieves the product metadata from IPFS using the IPNS address. 4. The system retrieves the Token ID and IPFS Hash chance log from the blockchain. 5. The system displays the project information, logistics history, and ownership history to consumer.
Alternate Flow	1. Invalid QR Code or QR Code Issues: i. The system notifies the customer: "Invalid QR code. Please try again." 2. IPFS Retrieval Failure: i. System retries the metadata retrieval from IPFS up to 3 times. ii. If retries fail, the system logs the issue and notifies the user.

Table 3.6: Use Case Specification for "Product Refund".

Use Case Name	**Product Refund**
Actor(s)	Consumer, Retailer
Description	A consumer requests a refund for a purchased product.
Preconditions	• The product token is currently owned by the consumer.
Postconditions	• The refund request is successfully processed. • The ownership of the product token is transferred back to the retailer. • The blockchain records the updated ownership transfer and refund transaction immutably.
Main Flow	1. Customer logs into the system and navigates to the “Product Refund” page. 2. Customer selects the product to refund and fill the refund reason. 3. The system validates the product refund request by checking the ownership of product token and refund eligibility. 4. The system approves the refund request if valid. 5. The system records the refund transaction and ownership transfer on the blockchain . 6. The system notifies both the retailer and consumer of the successful product refund process.
Alternate Flow	1. Refund Request Rejected: i. The system notifies the consumer of the rejection with a reason. 2. Blockchain Write Failure: i. The system logs the error and notifies the actor of the issue.

3.2.2.6 Activity Diagram

Activity Diagram illustrates the workflow or processes within a system, outlining the sequence of activities and decisions involved. It is helpful for understanding complex procedures or illustrating use cases as it displays the data and control flow between numerous steps. The Activity Diagram for each use case is shown below.

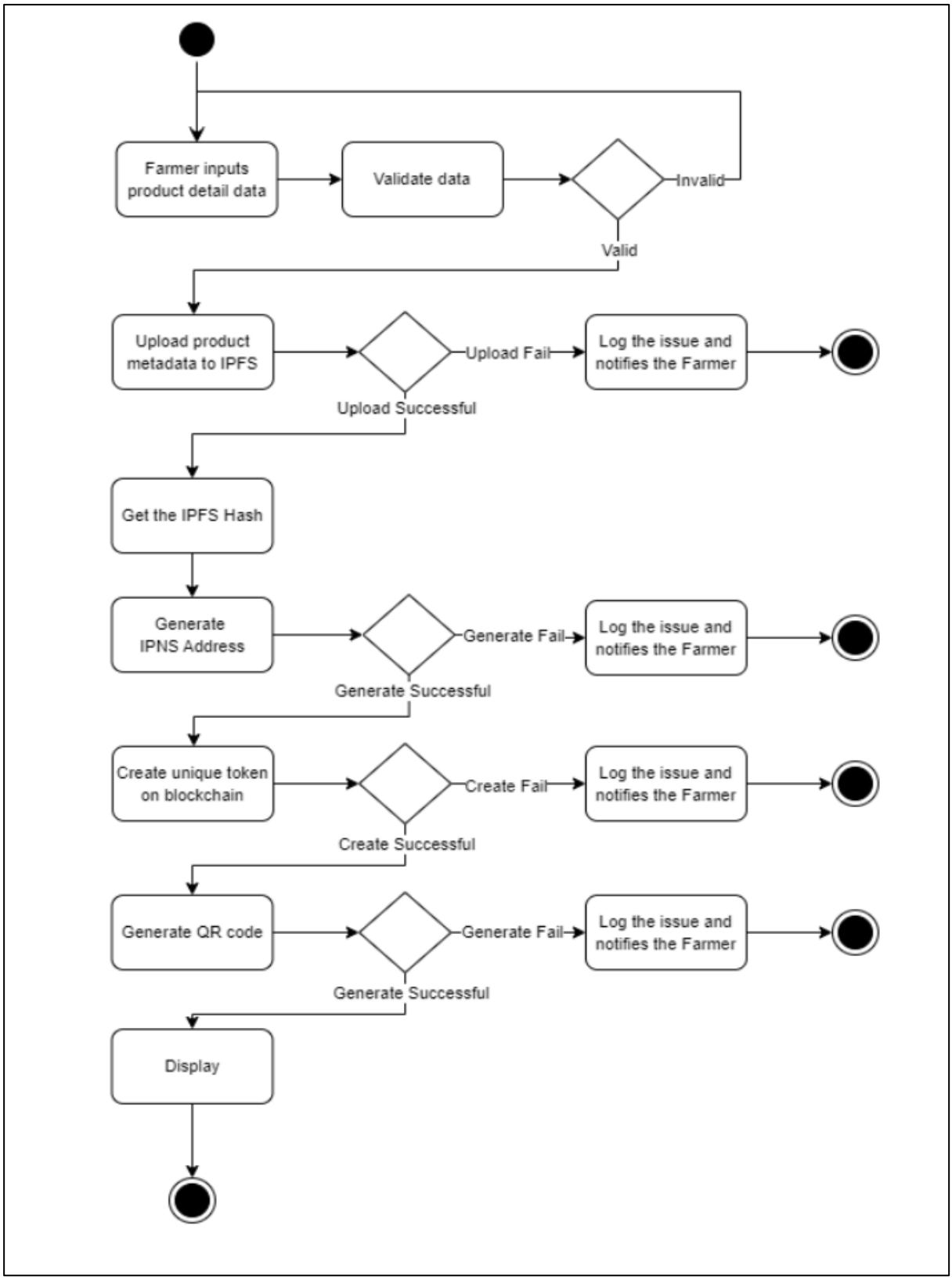


Figure 3.9: Activity Diagram for Use Case "Create Product Token".

Figure 3.9 describes activities which cover the flow of the use case “Create Product Token” and describe the steps that a farmer needs to follow in creating a token for a product. This starts with the farmer entering the product information which is then subsequently checked for validity. If the data is found to be incorrect, the system will ask the farmer enter the data again. If correct, the metadata of the product will then be added to the Interplanetary File System (IPFS). Once the system has verified that the upload is successful, the next step is for the system to get the IPFS hash. After this, an IPNS address will be created, the next step is for the system to check for success. If successful, a distinct token will be minted on the blockchain. Following successful creation, the product will have a QR code assigned to it. All steps can log an issue, and the farmer will be informed. The generated QR code is displayed as the last step in the process, which completes the workflow.

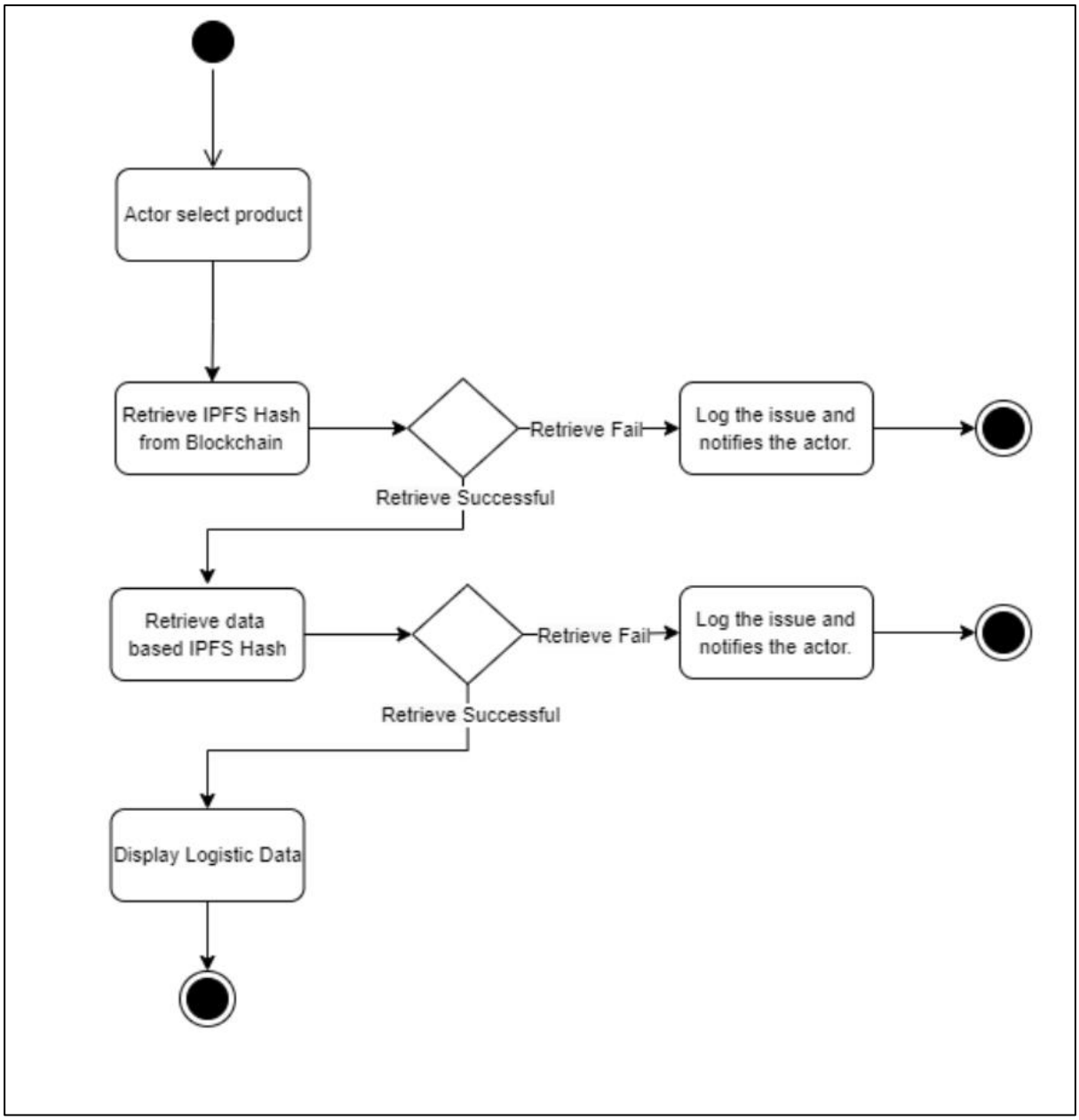


Figure 3.10: Activity Diagram for Use Case "Track Logistic Product".

Figure 3.10 depicting the use case “Track Logistic Product” demonstrates the several steps that an actor follows to track a product. The workflow begins with the actor selecting a product that they wish to track. The system then checks the blockchain to find the IPFS hash. If successful, the IPFS hash then enables the system to retrieve all the data associated with the IPFS hash. If any of the retrieval processes fails, the issue is recorded, and the actor is alerted.

In case the steps above are successful, the system is closed with the display of the logistic data of the product. The workflow, therefore, enables controlled retrieval and presentation of tracking data and incorporates provisions to deal with failures in the retrieval process.

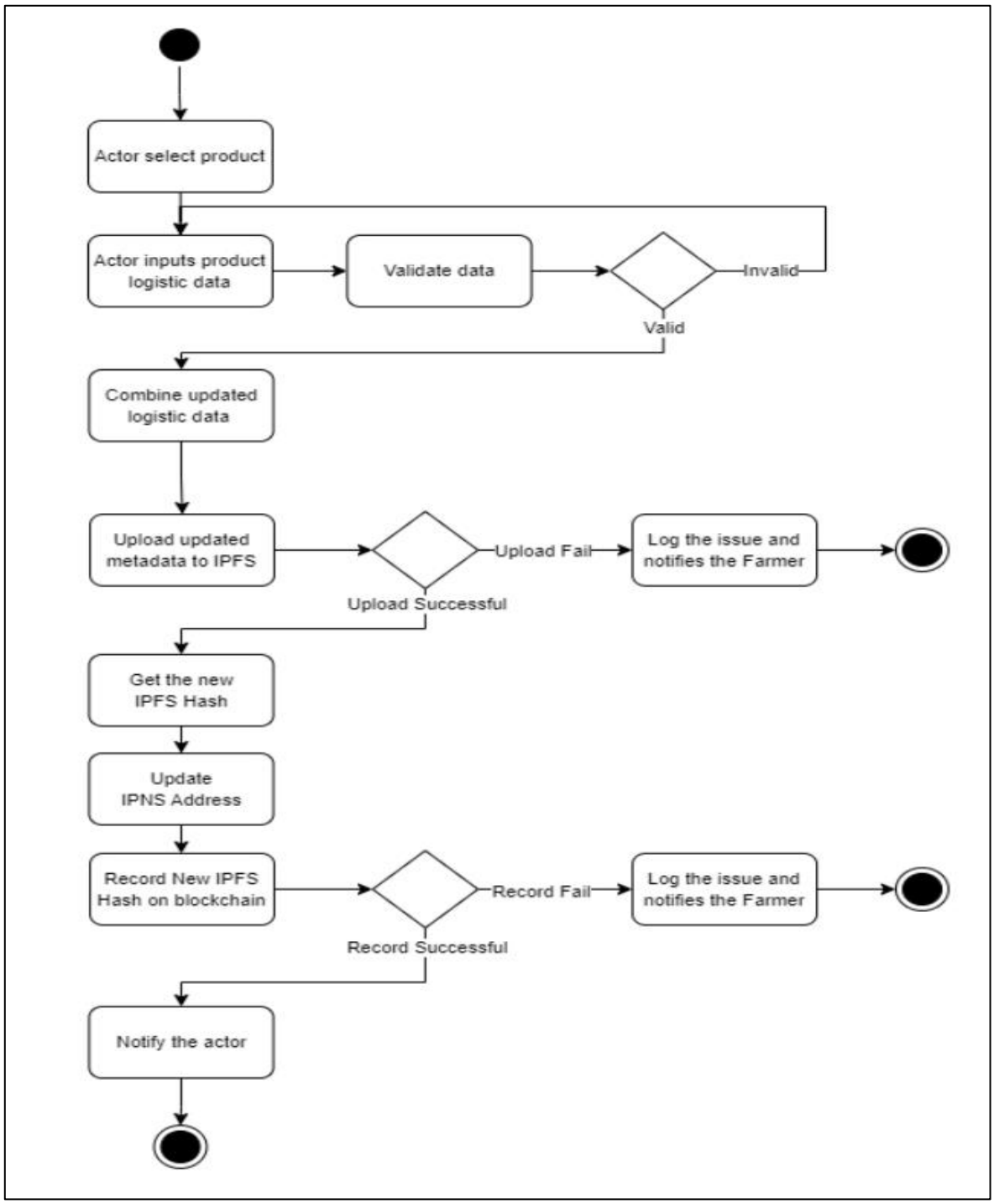


Figure 3.11: Activity Diagram for Use Case "Update Logistic Data".

Figure 3.11 shows the activity diagram for the use case “Update Logistic Data” and captures how the information related to a product is updated on the system. The process commences with the selection of a product for which an updated logistic data is provided. Following selection, the system performs validation on the data. Upon receiving invalid data, the actor is notified and the process terminates. When the data is valid, the updated logistic data is integrated and stored on Inter Planetary File System: IPFS. The new upload is successful, the system retrieves the new IPFS hash. The next step involves the system updating the IPNS address and registering the new IPFS hash into the blockchain. If some of the steps are not performed successfully, all the steps executed are logged and issue notifications sent to the actor. When successful, the actor is notified of the successful change. The process assures that the product's logistic data is maintained without any discrepancies, while at the same time enabling functionality to handle errors at any stage of the process.

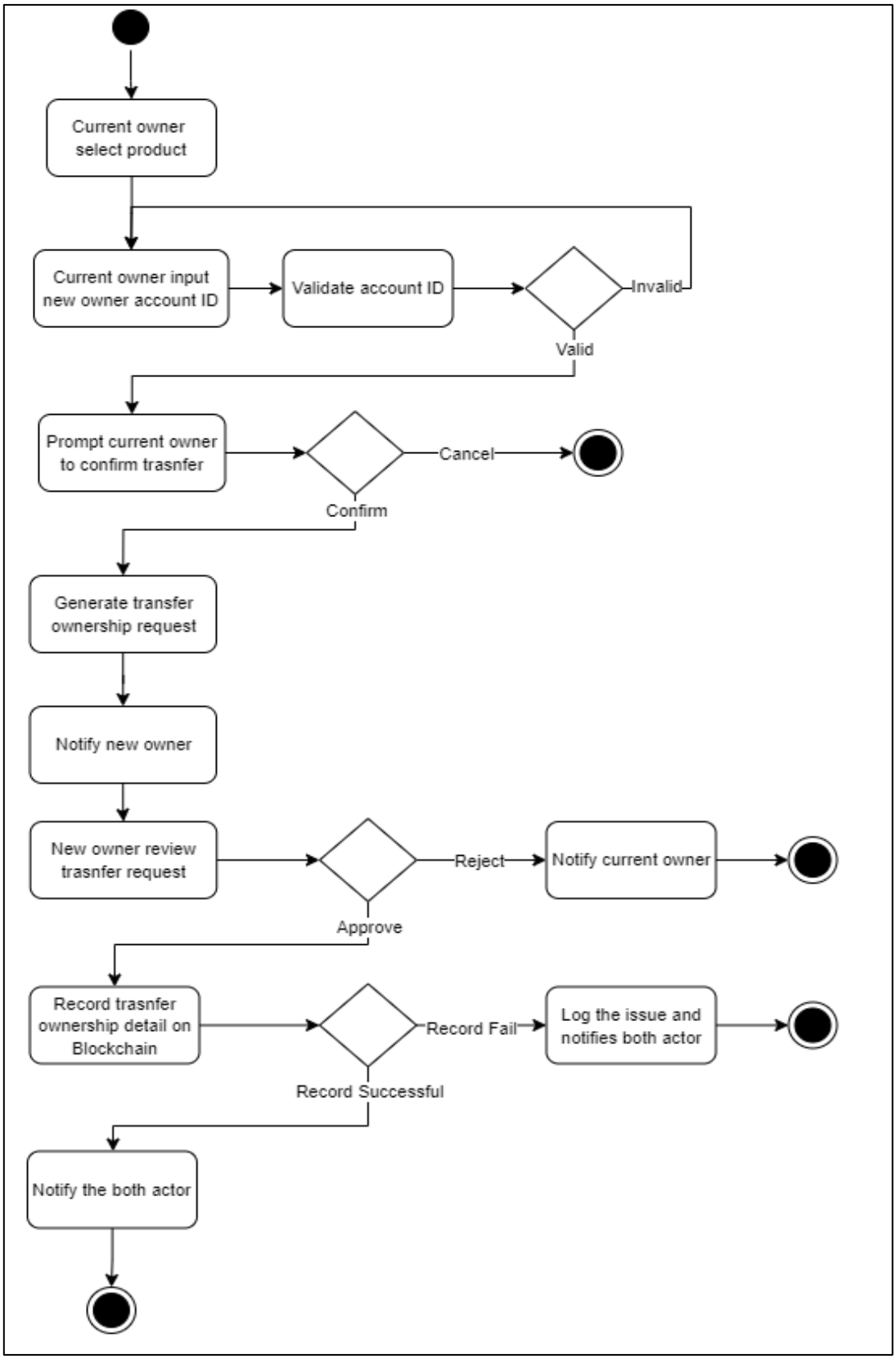


Figure 3.12: Activity Diagram for Use Case "Transfer Ownership".

Figure 3.12 displays the activity diagram for the use case “Transfer Ownership” and outlines the procedures for changing the ownership of a product from one owner to another. The process starts with the selection of a product by the current owner who should also enter the account ID of the new owner. The system then checks if the new owner’s ID is valid. If the validation is unsuccessful, the process terminates. Otherwise, the current owner is prompted to authorize the transfer. Once authorization is given, the system creates a transfer ownership request and alerts the new owner. The next step is for the new owner to assess the transfer request. Should the new owner opt not to accept the request, the current owner is alerted, and the process is terminated. Otherwise, once accepted, the system tries to register the transfer ownership transaction’s details on the blockchain. If this fails, the problem is recorded, and both users are alerted. If this step is successful, the last notification goes to both the current and new owners of the product thereby completing the ownership transfer cycle.

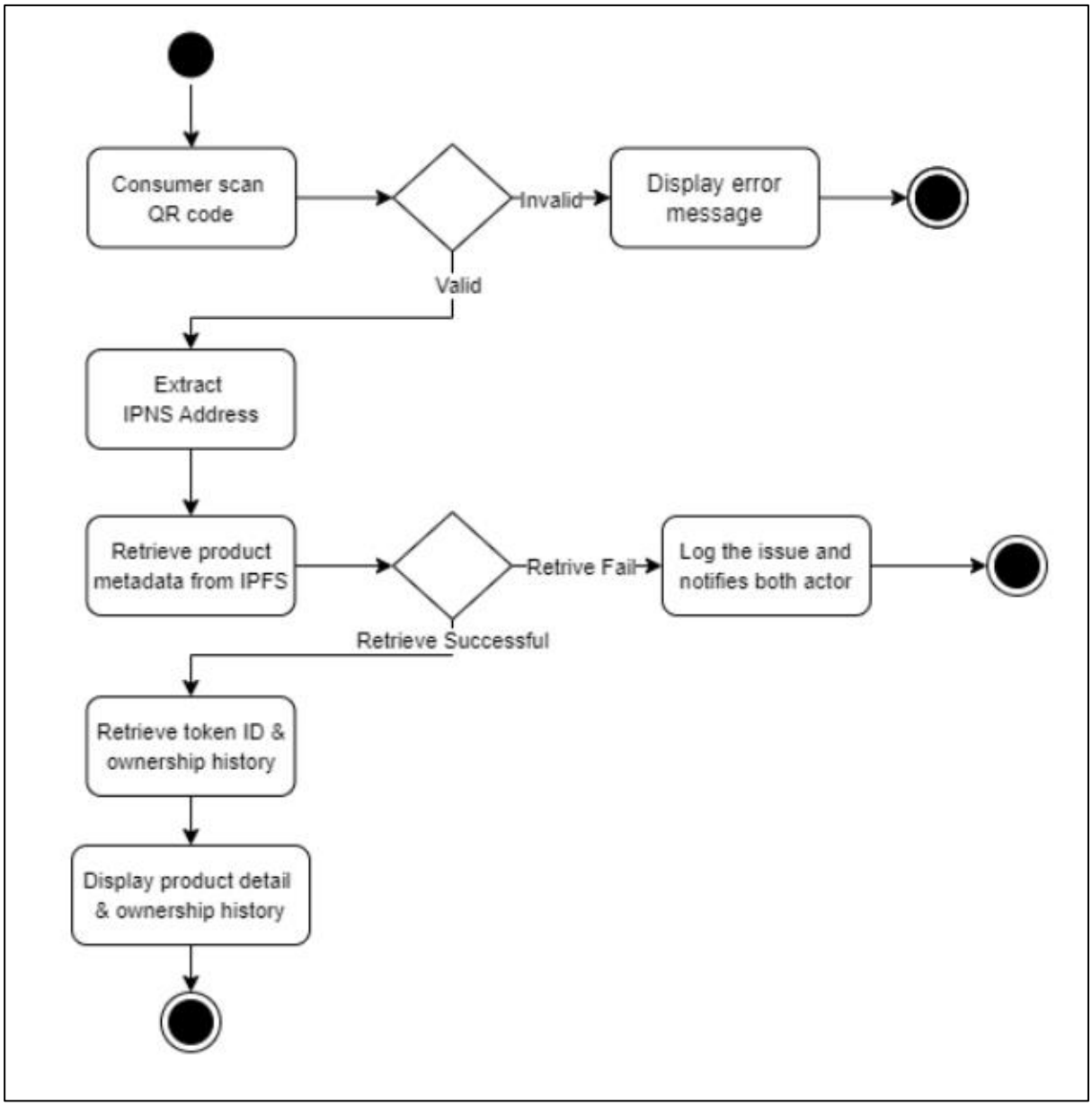


Figure 3.13: Activity Diagram for Use Case "View Product Detail and History".

Figure 3.13 shows the activity diagram for “Product Refund” and the sequence of actions performed by the consumer want to view the product information before purchase. The process starts with the consumer scanning the QR code provided on the product. In case the QR code is incorrect, an error will come up and the process terminated. If the QR code is correct, the system will fetch the IPNS address embedded in the QR code and will then attempt to get the product metadata stored in IPFS. If the metadata is not fetched, then the consumer along

with the appropriate actor will be informed so that the problem can be rectified. If the metadata fetching is successful, then the token ID with the ownership details of the product is fetched. In the end, the consumer is shown the product details with the ownership and refund details.

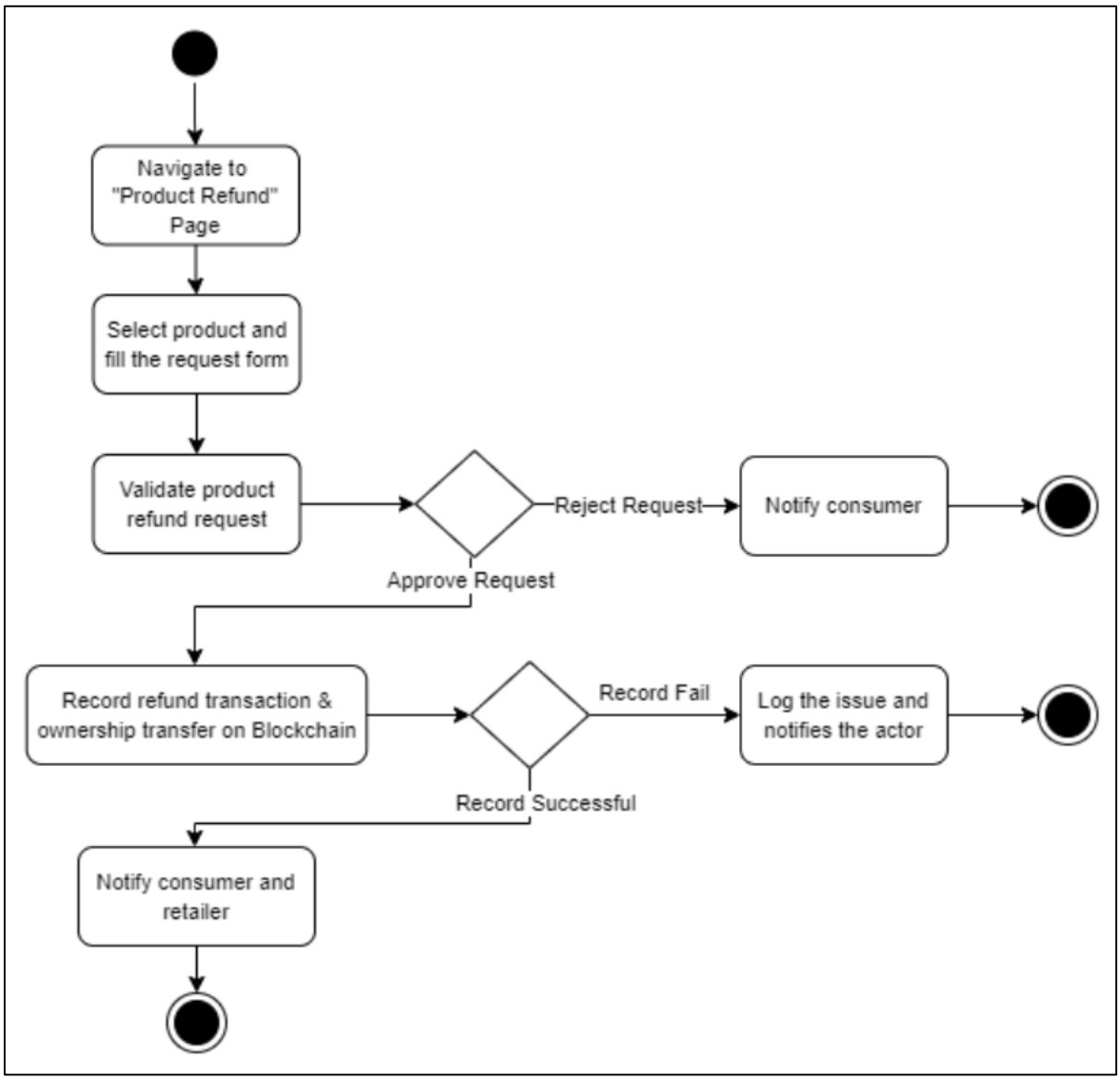


Figure 3.14: Activity Diagram for Use Case "Product Refund".

Figure 3.14 displays the activity diagram for the use case "Product Refund" and describes the steps involved when a consumer requests a product refund. The process begins when the consumer navigates to the "Product Refund" page and selects the product, filling out the refund request form. The system then validates the refund request. If the request is rejected,

the consumer is notified, and the process ends. If the request is approved, the system proceeds to record the refund transaction and ownership transfer on the blockchain. If the recording fails, the issue is logged, and the relevant actor is notified. If successful, both the consumer and the retailer are notified of the completed refund. This workflow ensures that the refund process is properly validated, recorded, and communicated to all relevant parties, with error handling in place at each step.

3.2.2.7 Sequence Diagram

Sequence Diagram represents how entities and system components interact with each other to complete a task. It records the order in which messages are transferred from actor to the system's components along with the information flow action into a coherent timeline. Following are the sequence diagrams for various identified use cases in the system, capturing all the interactions diligently required to meet the respective goals.

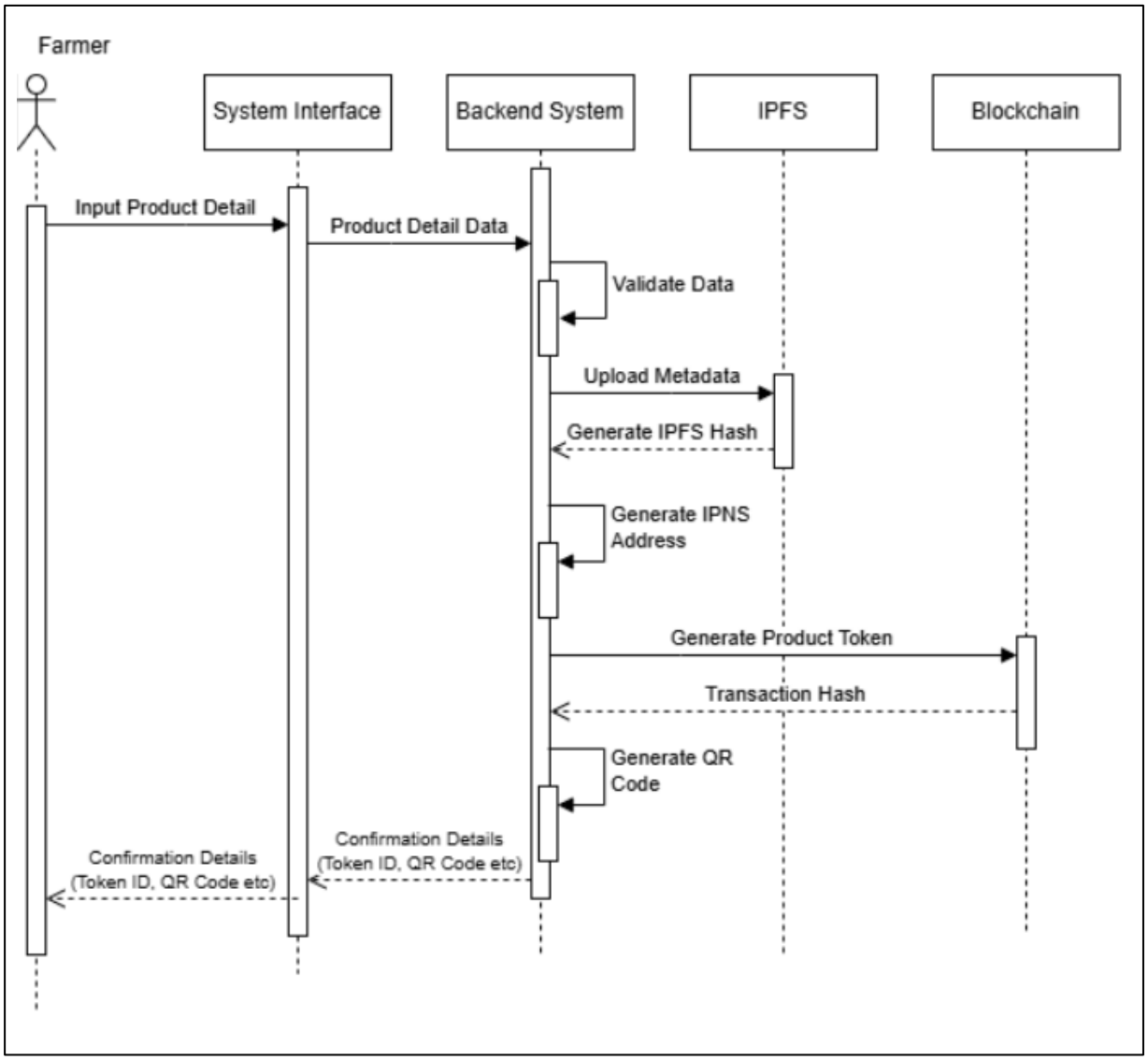


Figure 3.15: Sequence Diagram for Use Case "Create Product Token".

Figure 3.15 displays the sequence diagram for the use case "Create Product Token" and illustrates the interactions between the farmer, system interface, backend system, IPFS, and blockchain. The process begins with the farmer inputting product details into the system interface, which forwards the product detail data to the backend system. The backend system validates the data before uploading the metadata to IPFS. Once the metadata is uploaded, the system generates the IPFS hash and proceeds to create the IPNS address. The backend system then generates the product token on the blockchain and receives the transaction hash. Finally,

the backend system generates the QR code, which includes the token ID and other confirmation details. These confirmation details are sent back to the system interface and displayed to the farmer, completing the process of creating the product token. This diagram ensures that the entire product token creation process is properly executed, with each system component involved at different stages.

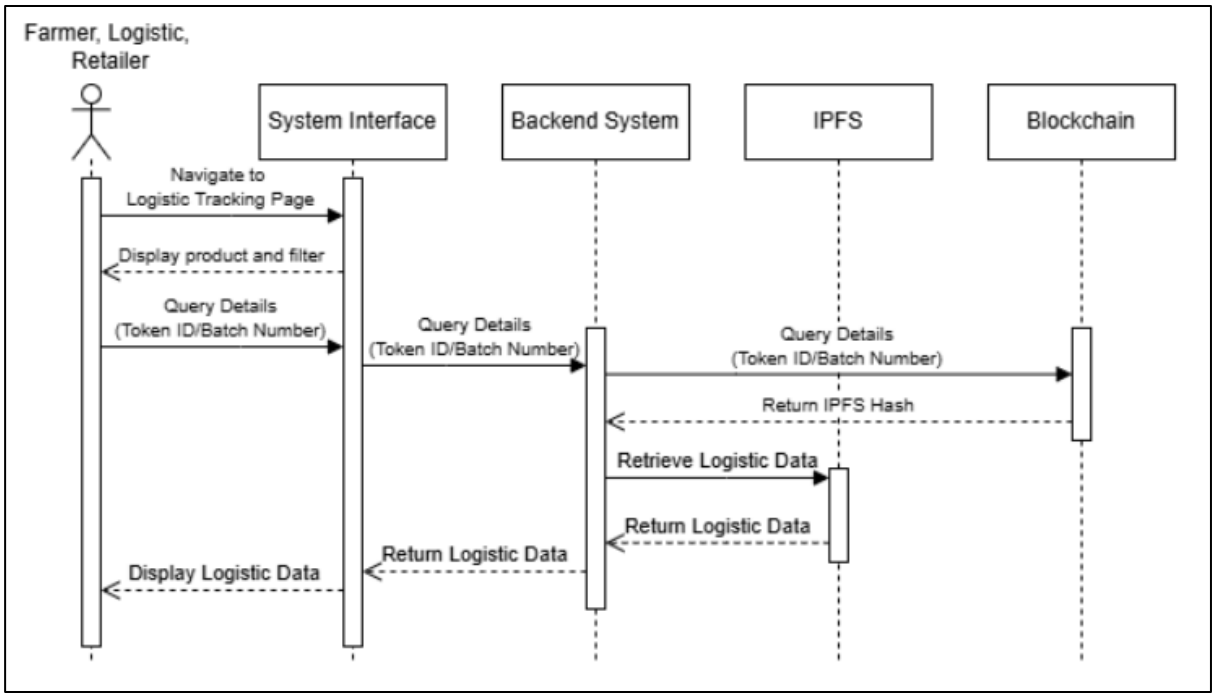


Figure 3.16: Sequence Diagram for Use Case "Track Logistic Product".

Figure 3.16 shows the sequence diagram for the use case "Track Logistic Product" and illustrates the process of tracking a product's logistic details, involving the farmer, logistics, retailer, system interface, backend system, IPFS, and blockchain. The process starts with the user (farmer, logistics, or retailer) navigating to the logistic tracking page on the system interface, where the product and filter options are displayed. The user queries the details, which include the token ID or batch number. This query is forwarded to the backend system, which further queries IPFS and the blockchain for the product's logistic data. IPFS returns the associated IPFS hash, and the backend system retrieves the logistic data linked to the hash.

Once the data is retrieved, it is returned to the backend system and displayed on the system interface, providing the user with the logistic information for the product. This sequence ensures that the logistic data is properly fetched from the blockchain and IPFS and displayed for the user.

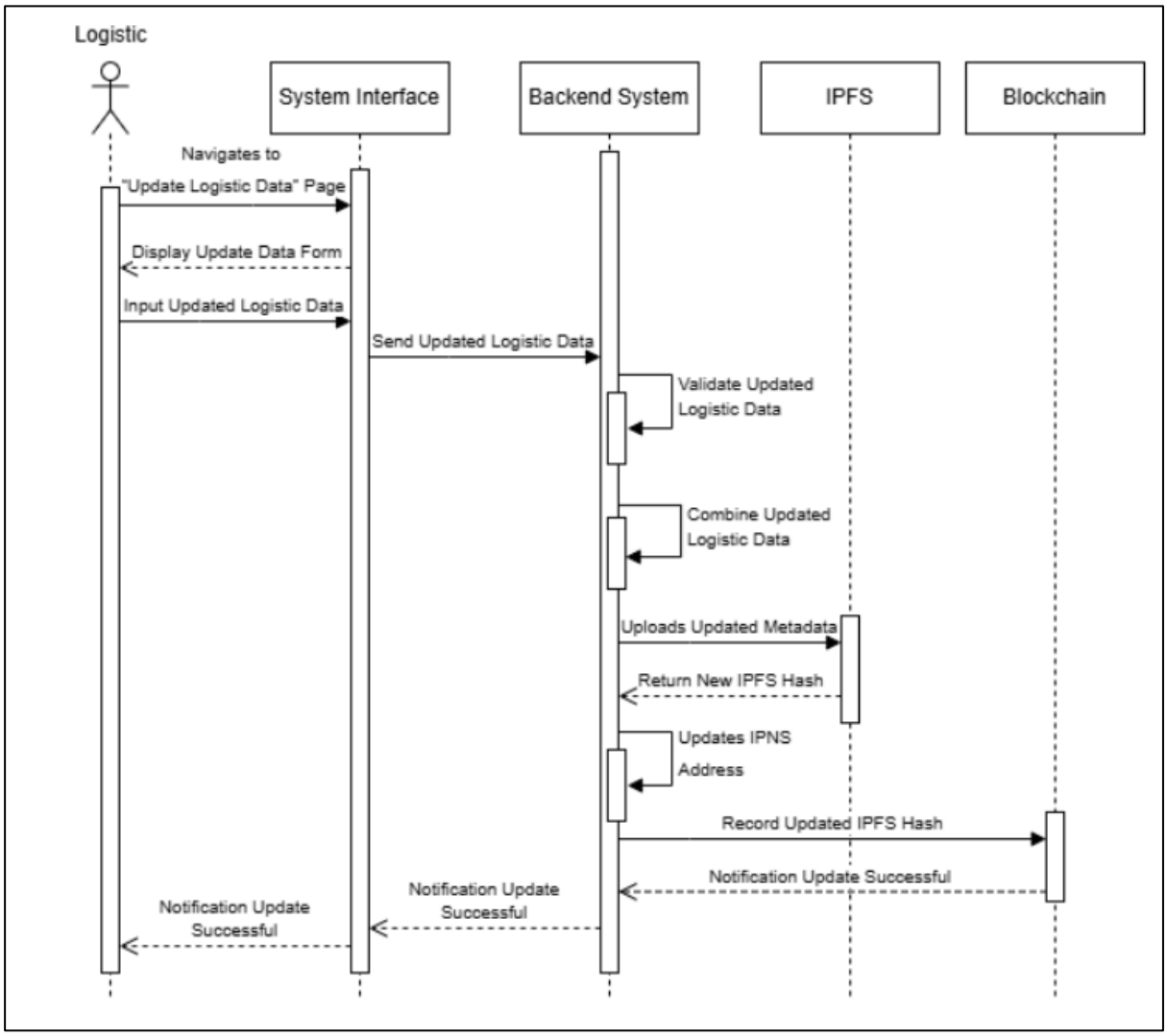


Figure 3.17: Sequence Diagram for Use Case "Update Logistic Data".

Figure 3.17 The sequence diagram for the use case "Update Logistic Data" outlines the process for updating the logistic details of a product. It begins with the logistic actor navigating to the "Update Logistic Data" page in the system interface, where the update data form is displayed. The logistic actor inputs the updated logistic data and sends it to the backend system.

The backend system validates the updated data and then combines the new logistic data. The system uploads the updated metadata to IPFS and returns the new IPFS hash. The backend system also updates the IPNS address, records the updated IPFS hash on the blockchain, and notifies the logistic actor that the update was successful. Finally, a successful notification update is sent to the system interface, completing the logistic data update process. This sequence ensures the seamless validation, updating, and recording of logistic data, while also ensuring notifications are sent to confirm successful actions.

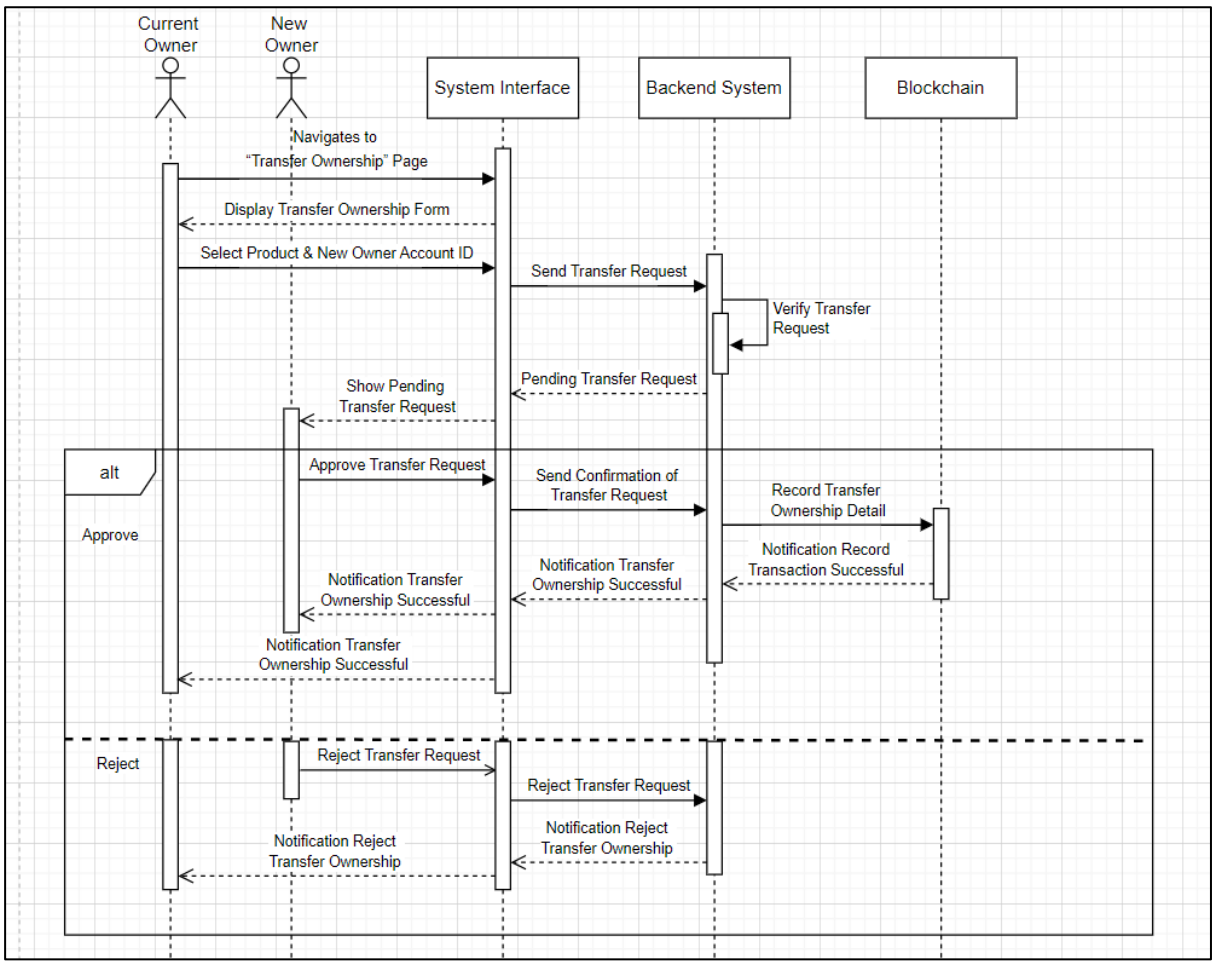


Figure 3.18: Sequence Diagram for Use Case "Transfer Ownership".

Figure 3.18 shows the sequence diagram for the use case "Transfer Ownership" and describes the process of transferring ownership of a product from the current owner to a new owner. The current owner starts by navigating to the "Transfer Ownership" page in the system

interface, where they select the product and input the new owner's account ID. The system then sends the transfer request to the backend system, which verifies the request. If the transfer request is approved by the new owner, the backend system sends confirmation to both the current and new owners, and the ownership details are recorded on the blockchain. A successful notification is then sent to both actors, confirming the transfer of ownership. If the new owner rejects the transfer, the backend system will notify both the current owner and the new owner of the rejection. The process ensures that the ownership transfer is properly verified, recorded, and communicated to the relevant parties.

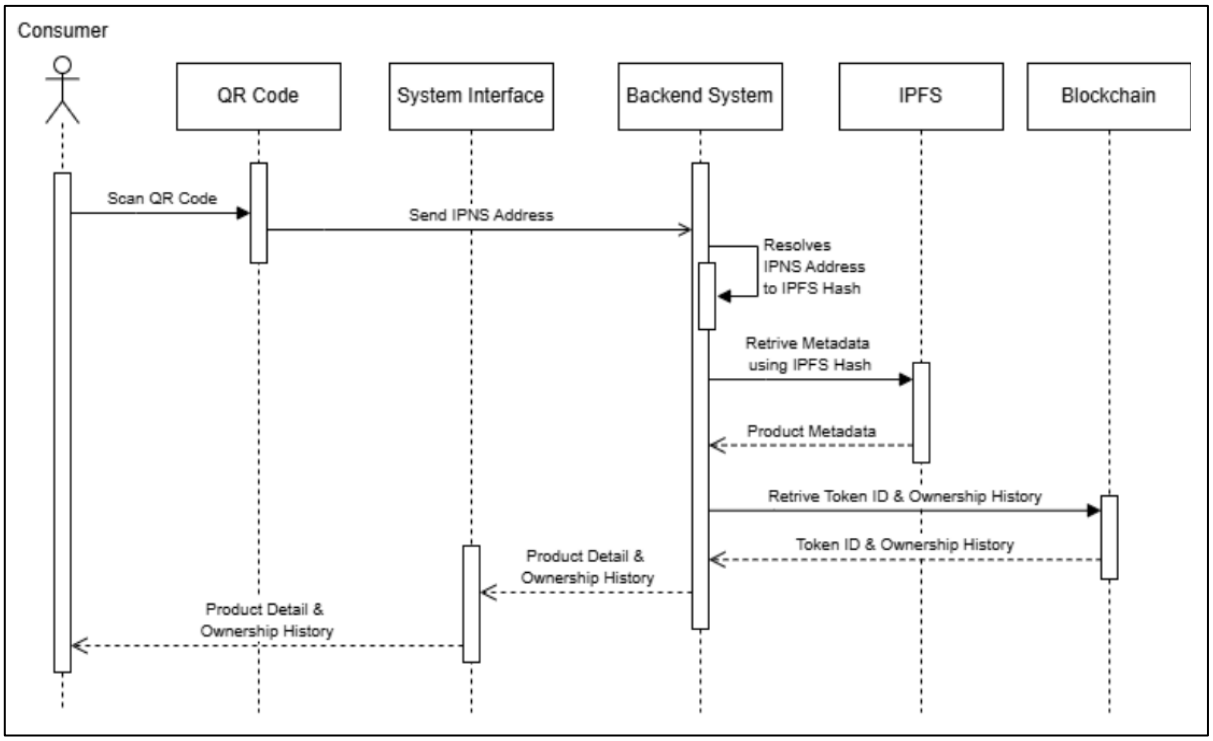


Figure 3.19: Sequence Diagram for Use Case "View Product Detail and History".

Figure 3.19 displays the sequence diagram for the use case "View Product Detail and History" and shows the interaction between the consumer, system interface, backend system, IPFS, and blockchain. The process starts when the consumer scans the product's QR code, which sends the IPNS address to the system interface. The backend system resolves this IPNS address to the corresponding IPFS hash. The backend system then retrieves the product

metadata using the IPFS hash. Once the metadata is fetched, the system also queries the blockchain to retrieve the product's token ID and ownership history. After gathering the necessary data, the system interface returns the product details and ownership history to the consumer, providing a comprehensive view of the product's information. This sequence ensures that consumers can access detailed and historical data about the product securely via the blockchain and IPFS.

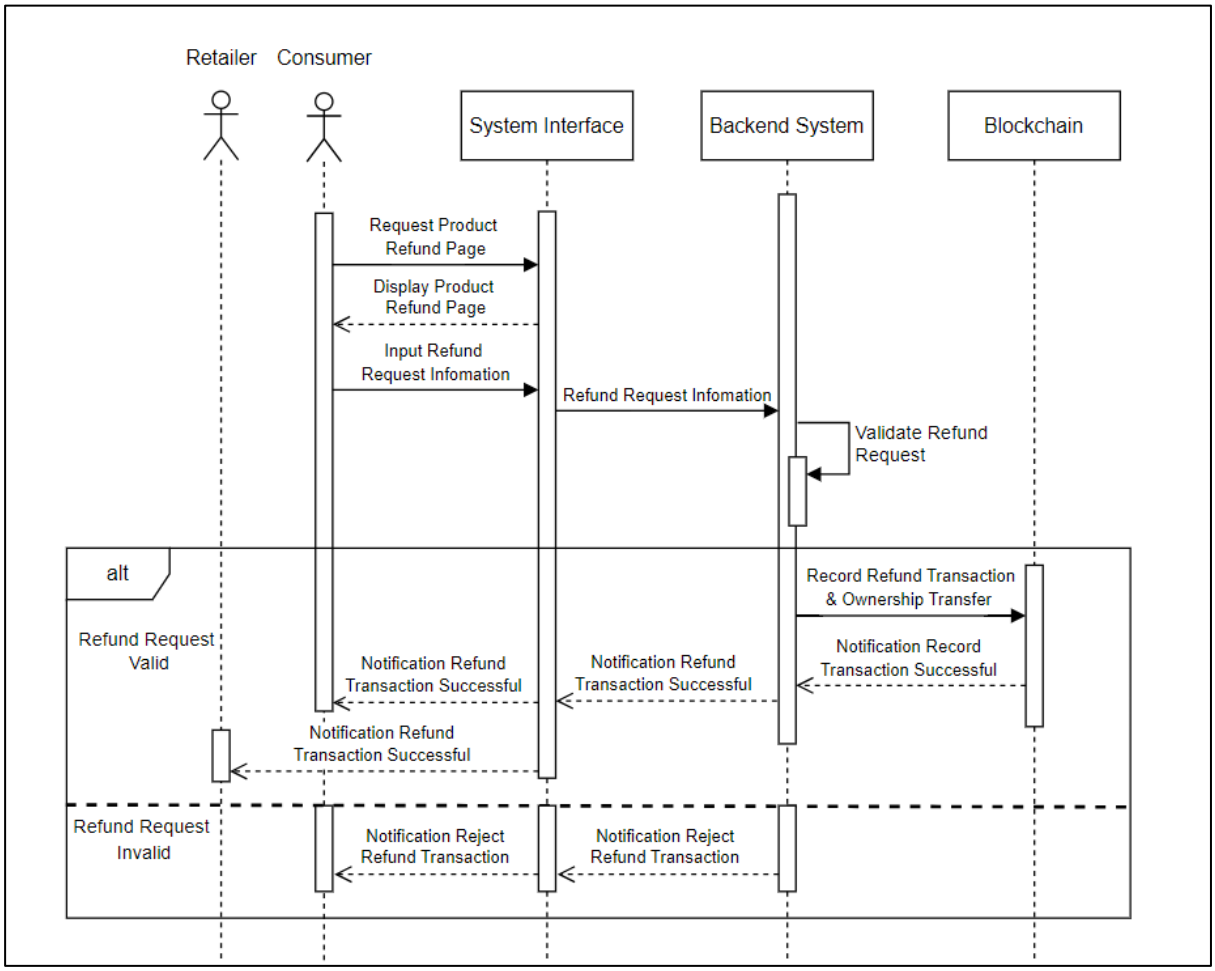


Figure 3.20: Sequence Diagram for Use Case "Product Refund".

Figure 3.20 shows the sequence diagram for the use case "Product Refund" and depicts the interaction between the retailer, consumer, system interface, backend system, and blockchain. The process begins when the consumer requests a product refund through the

system interface, which displays the refund request page. The consumer inputs their refund request information, which is sent to the backend system for validation.

If the refund request is valid, the backend system records the refund transaction and ownership transfer on the blockchain. Successful notifications are then sent to both the retailer and consumer, confirming the successful refund transaction and ownership transfer. If the refund request is invalid, the backend system sends a notification to both the retailer and consumer, rejecting the refund transaction. This diagram ensures that the refund process is validated, recorded, and communicated efficiently, with clear notifications for both the consumer and retailer.

3.3 Summary

In this chapter, the analysis and design of the proposed application is outlined. The system requirement is elicited by conduct the literature review to gather the best practice. Key elements of the system design, such as the context diagram, use case diagram, use case specifications, activity diagram, and sequence diagram, are presented. These diagrams and specifications serve as the groundwork for the subsequent implementation phase, ensuring a clear and structured approach to system development.

CHAPTER 4 IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter will discuss the system implementation and testing of the proposed solution. The implementation section outlines the development process, detailing the integration of key technologies such as Node.js for the backend service, IPFS Filebase for decentralized file storage, Hardhat for smart contract development, and MetaMask for user interaction with the blockchain. It will also describe how each component was configured and integrated to ensure smooth and efficient operation within the overall system.

In the testing section, the focus will be on the tests designed to ensure the system meets the specified requirements, performs as expected under varying conditions, and provides a secure and efficient environment for users to interact with the blockchain.

4.2 System Implementation

This section discussed how the proposed system will be developed. The installation process and setting of NodeJS, Filebase, Hardhat Environment, and MetaMask will be discussed. The purpose of this chapter is to provide an overview of the implementation process and a detailed record of the actions taken to ensure the successful development of proposed system.

4.2.1 Installation

In this section, the installation steps for the core components of the system, including Node.js, IPFS Filebase, Hardhat, and MetaMask, are outlined. Each component is essential for building and interacting with the proposed decentralized application (DApp) and ensuring smooth functionality.

4.2.1.1 NodeJS

In the proposed system, Node.js is utilized to build the backend service. Node.js is a JavaScript runtime environment based on the Chrome V8 engine, enabling JavaScript to run on the server side, making it ideal for high-performance network applications. To set up the Node.js development environment on the local machine, the appropriate version for the operating system must be downloaded and installed from the official Node.js website. The installation process is straightforward. The official Node.js website, as shown in Figure 4.1, should be visited, and the correct version for the operating system, such as Windows, should be selected. The installation package for Windows is in the .msi format, which can be downloaded and installed accordingly.

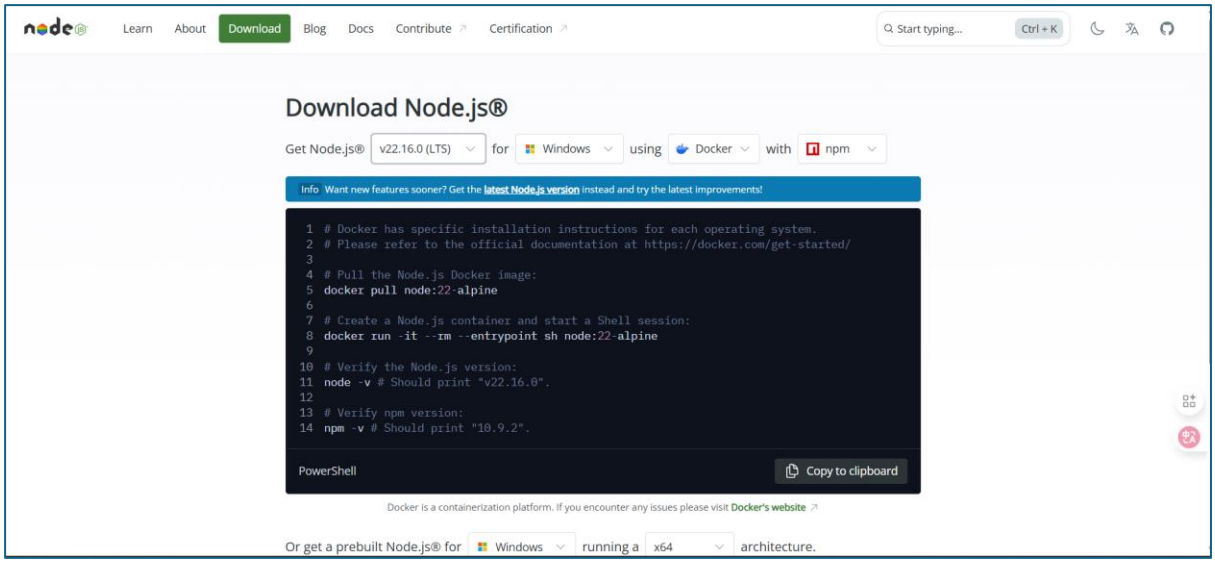


Figure 4.1: Official Node.js Website.

Once the installation is complete, the success of the Node.js installation can be verified via the command line. By opening the terminal and entering the command, as shown in Figure 4.2, the versions of Node.js and npm (Node.js package manager) will be displayed.

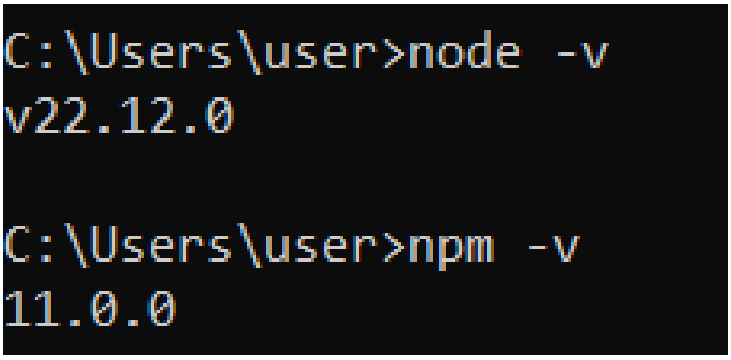


Figure 4.2: Command to Check the Version.

After that, the next step is to initialize a new Node.js project. This is done by navigating to the directory where the project will be stored and executing the commands, as shown in Figure 4.3. These commands will create a new project directory and generate a package.json file within that directory. The package.json file is the core file for Node.js project management, containing basic information about the project and listing all the dependencies and libraries used in the project.

```
mkdir blockchain-based-food-supply-chain
cd blockchain-based-food-supply-chain
npm init -y
```

Figure 4.3: Command to Create a New Project Directory

After setting up the basic environment, the next step is to install various third-party dependencies using npm install. For the backend service of the proposed system, the Express framework is chosen. Therefore, the "npm install express" command is executed to install Express. This will install Express and add it to the project's dependencies in the package.json file. In the root directory of the project, an index.js file is typically created. The index.js file serves as the entry point for the application, typically containing code for setting up the server, configuring routes, initializing middleware, and other essential functions. In this file, the

behaviour of the Express application is defined, such as listening on a specific port and handling incoming requests.

To improve development efficiency, the nodemon tool is also installed. Nodemon is a useful tool that automatically monitors code changes and restarts the server when code is modified. This eliminates the need for manually restarting the server after each code change, saving time and effort. To install nodemon, the npm install --save-dev nodemon command is used. Once installed, the scripts section in the package.json file is updated with the configuration, as shown in Figure 4.4. This allows the server to be started by running npm run start. Whenever the code is modified and saved, nodemon will automatically restart the server, ensuring a more efficient development process.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "nodemon index.js"
},
```

Figure 4.4: Configuration for Setting Up Nodemon.

4.2.1.2 IPFS Filebase

IPFS Filebase is a file storage platform based on IPFS technology, providing simple and user-friendly interfaces and features that help developers store and manage files on a decentralized storage network. Filebase simplifies the process of using IPFS by offering a unified API and storage interface, enabling developers to easily upload files to the IPFS network and retrieve them.

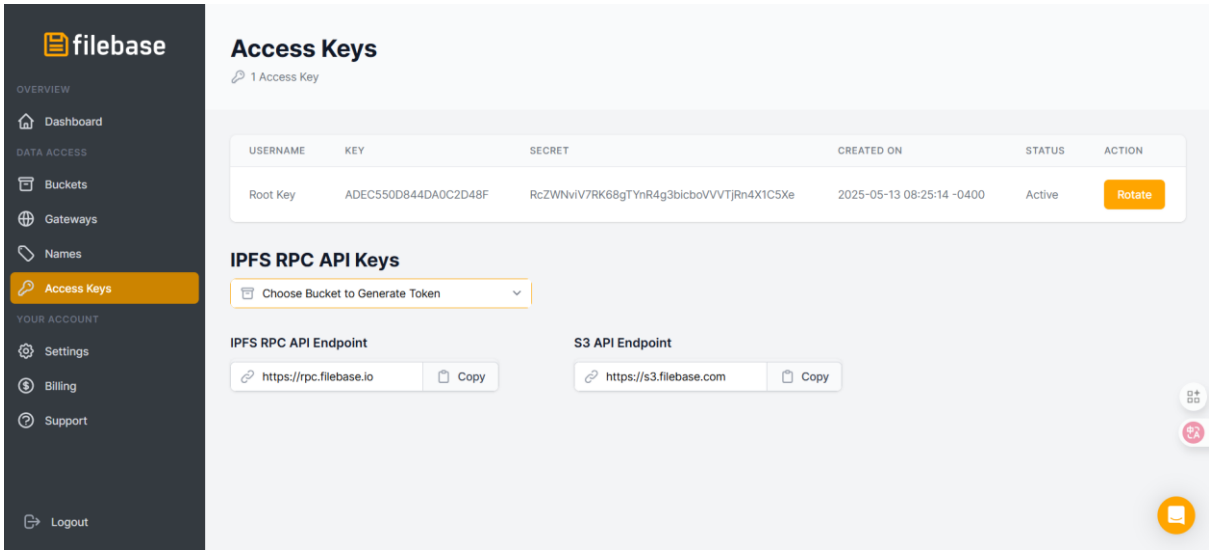


Figure 4.5: The API Keys Management Page for Filebase

To use the services provided by Filebase, users must first register and obtain an API key, as shown in Figure 4.5. Once registered, users can use the Filebase API to upload files to the IPFS network and generate a unique content identifier (CID). These CIDs are hash values of the file content, ensuring the file's uniqueness and immutability. Users can access the stored files through the CID, ensuring that the files always remain intact and verifiable.

4.2.1.3 Hardhat Environment

Hardhat is a development environment designed for Ethereum, aimed at simplifying the process of building, testing, and deploying smart contracts, particularly for Ethereum-compatible blockchains. It provides developers with a flexible and extensible framework, making it easier to develop decentralized applications, especially when interacting with the Ethereum network.

In this project, Hardhat will be used to efficiently develop and deploy smart contracts. Hardhat provides a local Ethereum network, which is ideal for testing and debugging smart contracts before deploying them to a live network. This local network is fast, stable, and allows

developers to test contracts without using real Ether. Additionally, Hardhat provides several test accounts for developers to simulate transactions and test contract functionality.

To set up the Hardhat environment, the "npx hardhat" command must be run within a Node.js project. This command helps automatically generate the necessary project structure and configuration files. Once the Hardhat environment is set up, developers can write Solidity smart contracts in the contracts directory and use Hardhat's local network for quick testing. In the hardhat.config.js file, the network configuration should be set up as shown in Figure 4.6, where the appropriate network structure is added to configure the local and Hardhat networks for testing purposes.

```
module.exports = {
  solidity: "0.8.28",
  networks: {
    hardhat: {
      chainId: 1337,
    },
    localhost: {
      url: process.env.LOCALHOST_URL,
      chainId: 1337,
    },
  },
};      You, last month • Initial commit
```

Figure 4.6: Configuration for setting hardhat network.

4.2.1.4 Metamask

MetaMask is a popular cryptocurrency wallet and gateway to blockchain applications, specifically for interacting with Ethereum and other Ethereum-compatible blockchains. It allows users to manage their crypto assets and interact with decentralized applications directly from their web browser. MetaMask is available as a browser extension (for Chrome, Firefox, Brave, and Edge) as well as a mobile app for both Android and iOS. In the proposed system, MetaMask will play a crucial role in the interaction between users and the blockchain. It acts

as the bridge that allows users to sign transactions and interact with smart contracts deployed on the Ethereum network. Through MetaMask, users can connect to the blockchain without needing to run a full node, simplifying access to decentralized applications.

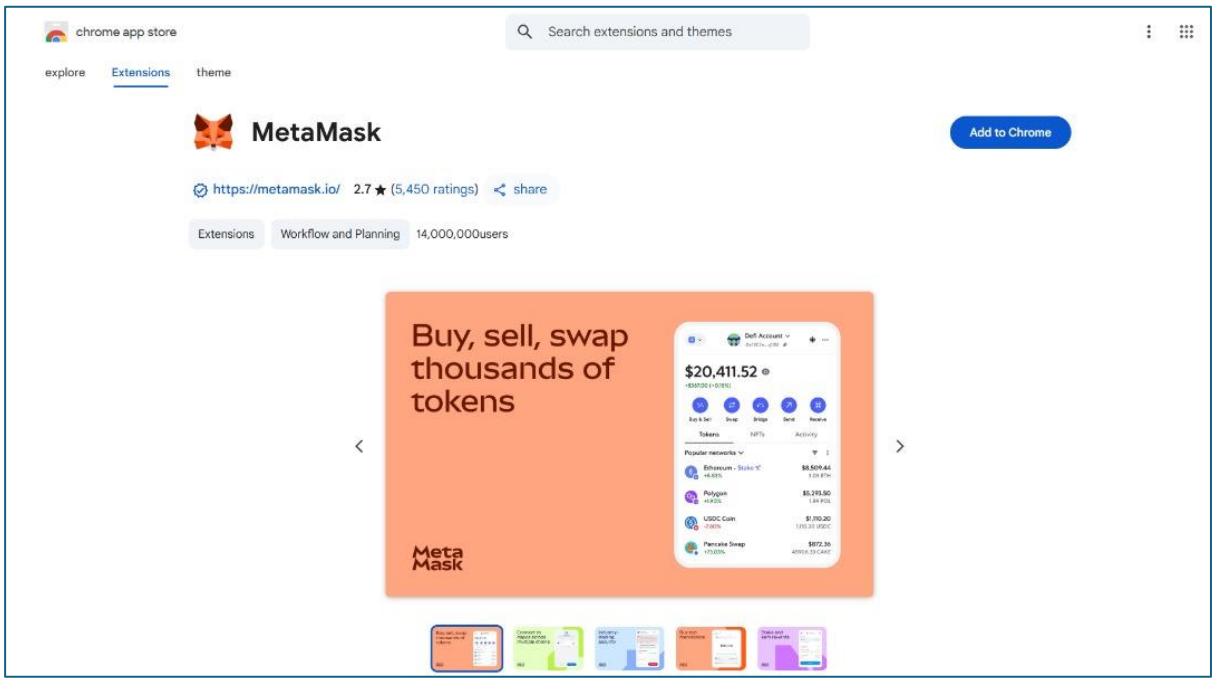


Figure 4.7: Installation of MetaMask on Google Chrome Extension.

In this project, MetaMask was implemented as the cryptocurrency wallet and gateway for interacting with the Ethereum blockchain and decentralized applications. To integrate MetaMask into the system, the MetaMask extension was installed on Chrome. The process involved visiting the Chrome Web Store, searching for MetaMask, and clicking "Add to Chrome" to install it, as shown in Figure 4.7. Once installed, the MetaMask icon appeared in the browser's extension bar, allowing easy access to the wallet directly from the browser. Additionally, the MetaMask mobile app was installed on iOS devices, which can be downloaded from the App Store. This provided flexibility to manage the wallet and interact with the system both on desktop and mobile.

To connect MetaMask to the localhost network, several steps were followed to ensure seamless interaction between MetaMask and the Hardhat local network, which was used for testing and development. First, the Hardhat local Ethereum network was started by running the "npx hardhat node" command in the terminal.

Figure 4.8: Terminal After Running the Command

This command launched a local Ethereum node, and the terminal displayed the RPC URL (http://127.0.0.1:8545), as shown in Figure 4.8, which was used to connect MetaMask to the local network. Next, MetaMask was opened, and the settings menu was accessed. In the "Networks" section, the "Add Network" option was selected to configure the connection. As Figure 4.9, the following details were entered:

- **Network Name**: "Localhost 8545"
- **RPC URL**: http://127.0.0.1:8545 (the URL provided by the running Hardhat node)
- **Chain ID**: 1337 (the default chain ID for Hardhat's local network)

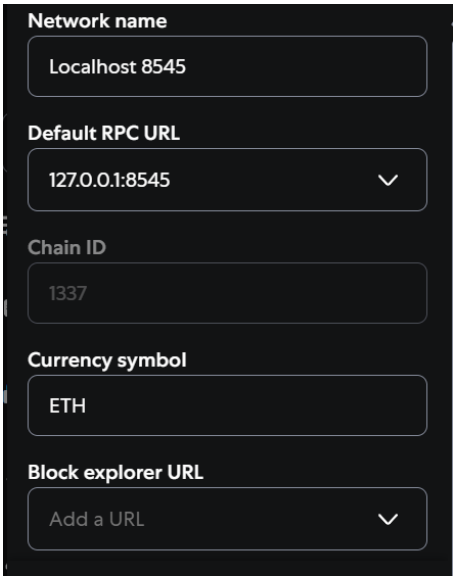


Figure 4.9: Network Configuration on MetaMask to Add Localhost Network.

After saving the configuration, MetaMask was successfully connected to the Hardhat local network. This allowed for interaction with the Ethereum blockchain using MetaMask, enabling testing of transactions, deployment of smart contracts, and simulating real-world interactions in a local and secure environment before moving to the live network.

This setup ensured seamless integration of MetaMask into the project, providing a reliable and secure method for interacting with smart contracts and conducting blockchain-related operations.

4.2.2 The Developed System

4.2.2.1 Login

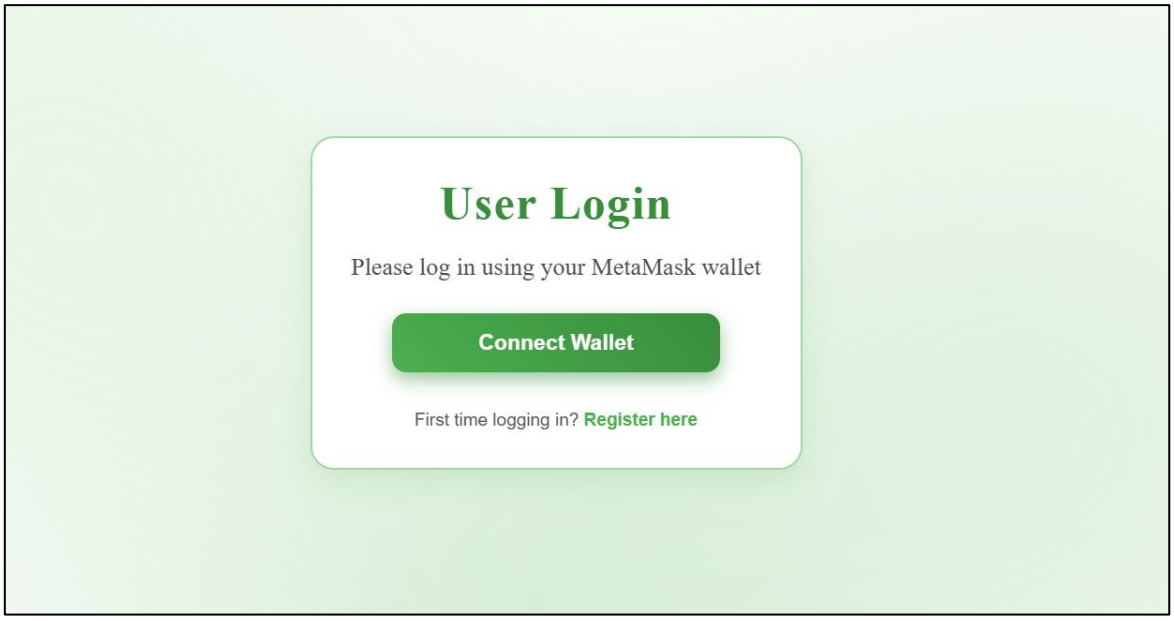


Figure 4.10: The Login Page.

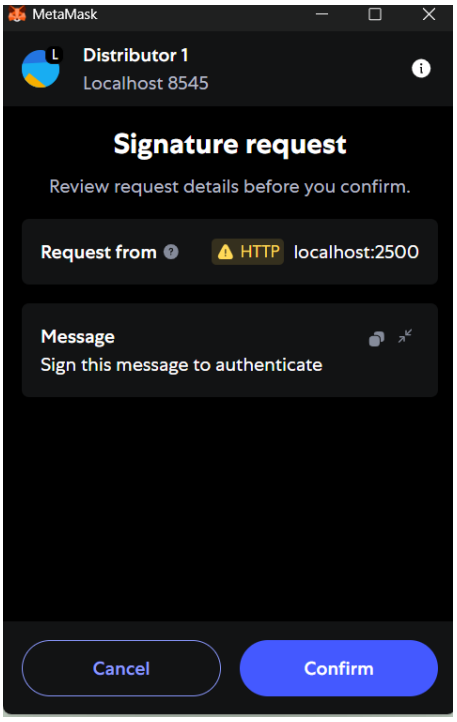


Figure 4.11: MetaMask pop-up window.

Figure 4.10 shows the login page. When the user clicks the login button, the system will automatically check whether the currently selected MetaMask account is already registered. If not, a message will appear asking the user to register first. If the account is already registered, MetaMask will pop up a window as shown in Figure 4.11, containing an authentication message for the user to sign. After clicking the Confirm button, the user will log in to the system successfully.

4.2.2.2 Register

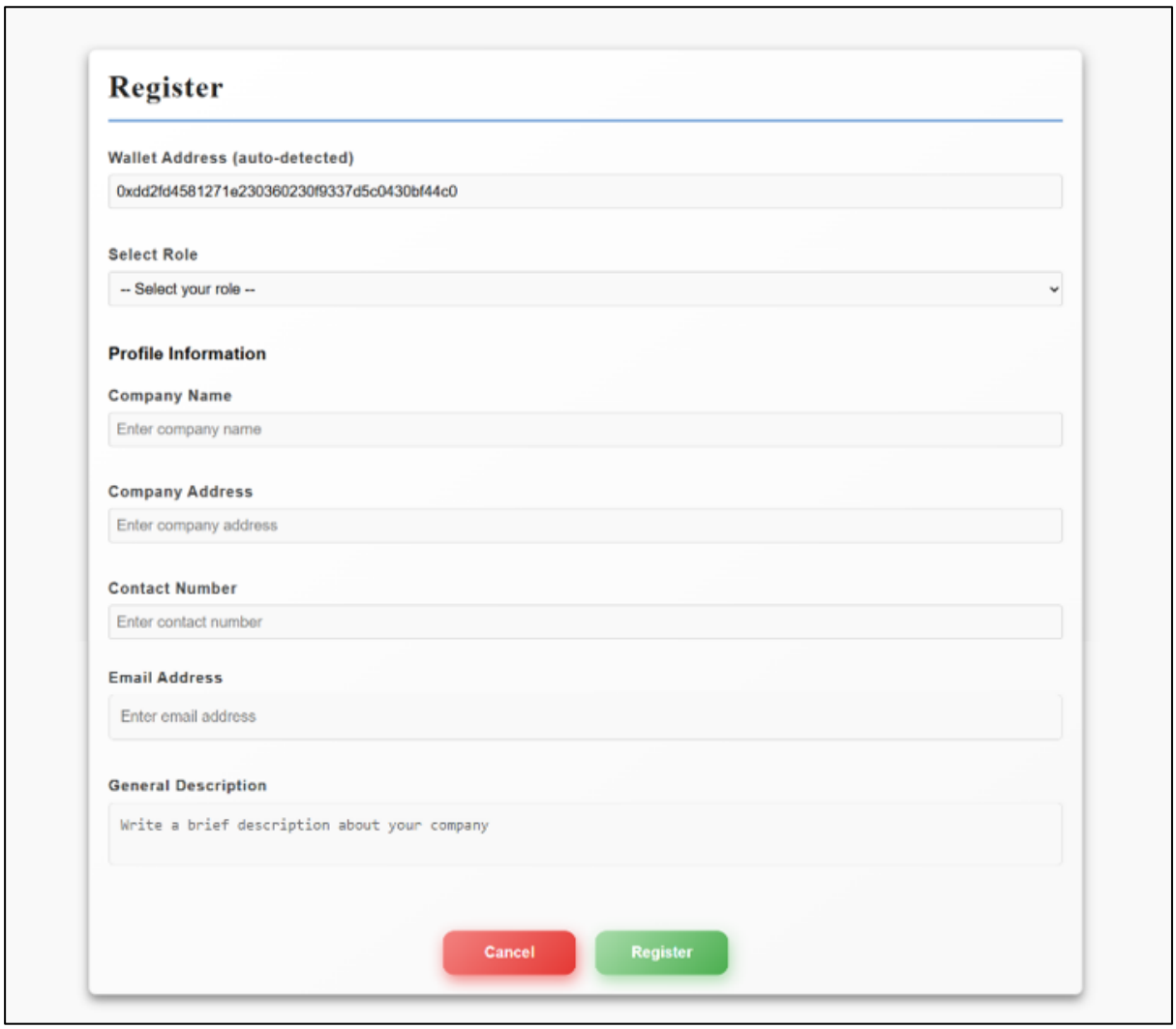


Figure 4.12: The Registration Form Page

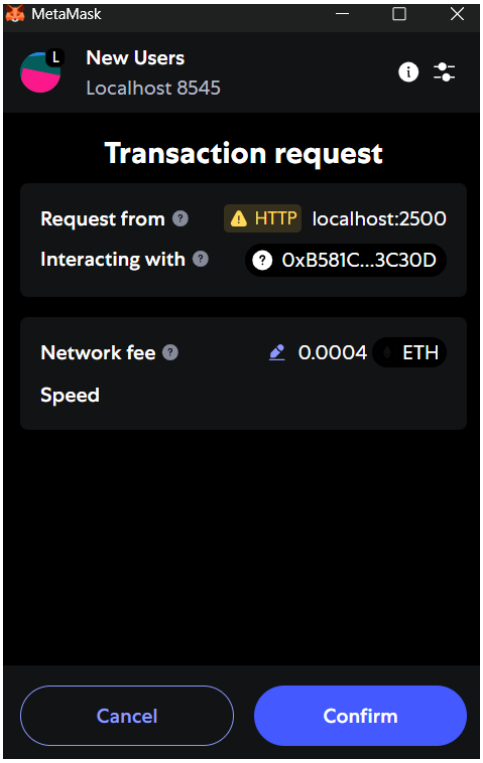


Figure 4.13: MetaMask pop-up window.

Figure 4.12 shows the registration form page. The Wallet Address input field automatically detects the currently selected MetaMask account. The user needs to select the role they want to apply for. If the selected role is Consumer, the profile information section will automatically update to display fields for Name, Address, Contact Number, and Email Address. For the roles Farmer, Distributor, or Retailer, the profile section will instead display Company Name, Company Address, Contact Number, and Email Address.

After clicking the Register button, a MetaMask pop-up window will appear, as shown in Figure 4.13. This pop-up requests the user to confirm the transaction, which will store the registration data on the blockchain. Once the user confirms, the system will display a success message and redirect them to the login page.

4.2.2.3 Main Page for Role Farmer

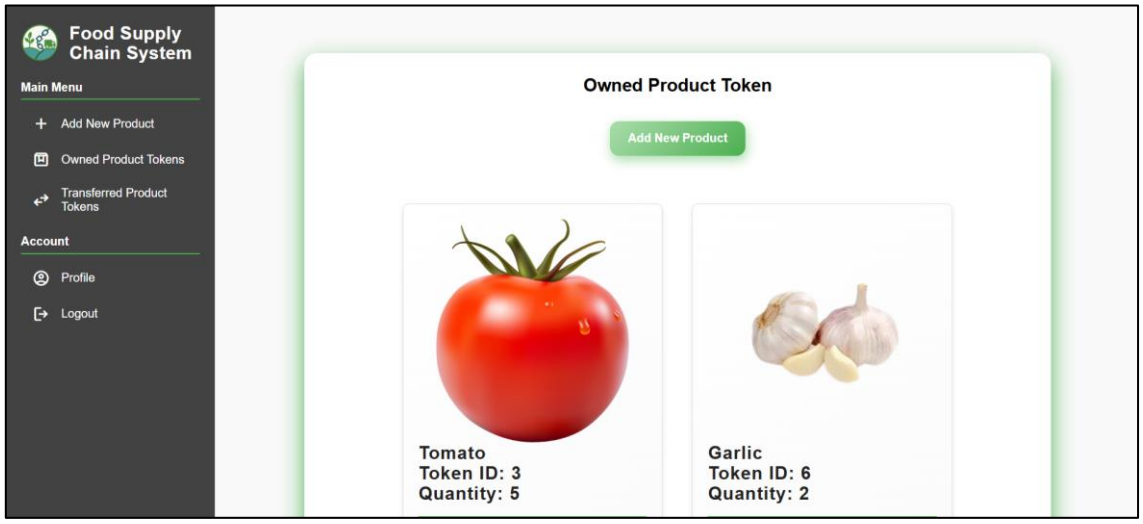


Figure 4.14 Main Page for Role Farmer.

After a successful login, users with the Farmer role will be directed to the farmer main page, as shown in Figure 4.14. This page provides access to features specifically designed for farmers. The interface includes a navigation bar that allows the user to add new product tokens, view all owned products, and track transferred product tokens in the supply chain.

4.2.2.4 Owned Product Token Page

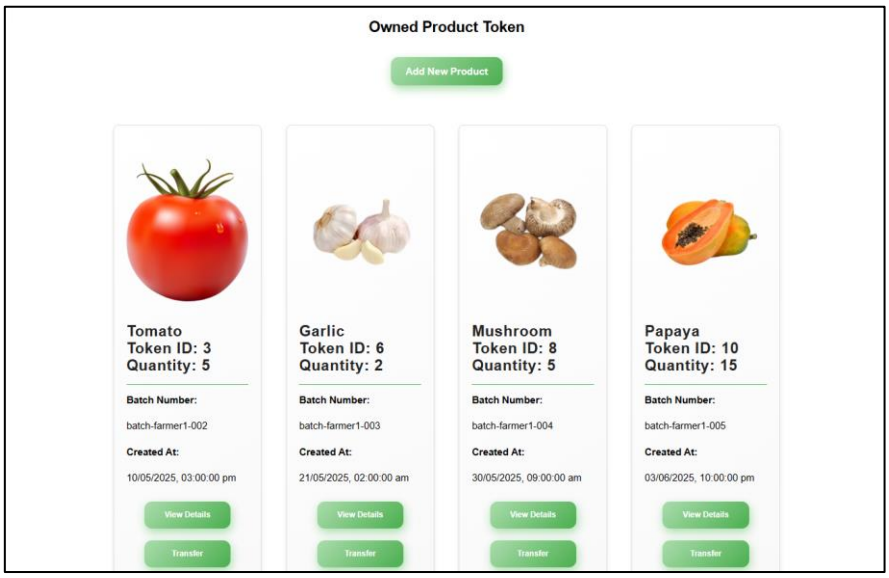


Figure 4.15: The Owned Product Token Card Page

Figure 4.15 shows the Owned Product Token Card page. Each owned product is displayed in a card format, showing essential details such as the product name, batch number, and date of creation. Each card includes a Transfer button, allowing the farmer to transfer the product to the next party in the supply chain. There is also a View button that allows the user to see the full information of the product token.

4.2.2.5 Create New Product Token

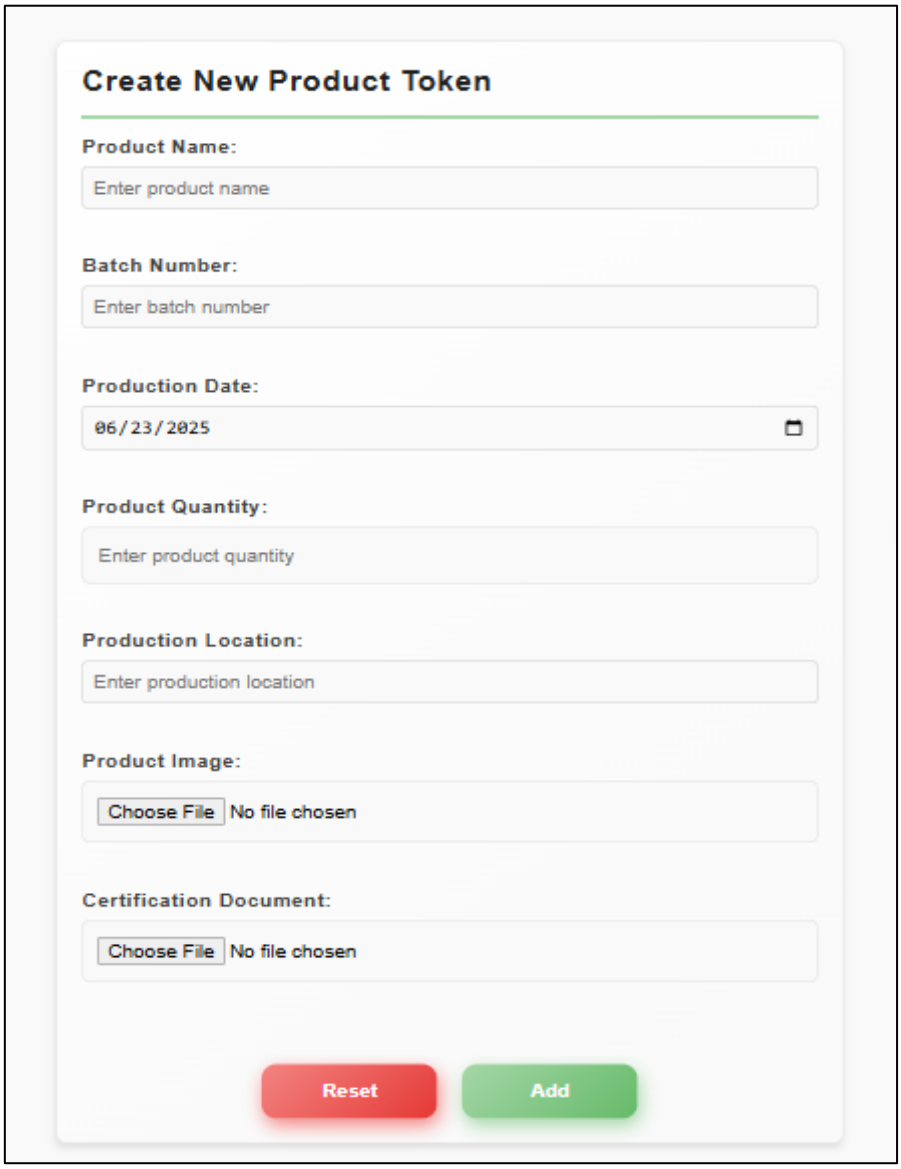


Figure 4.16: The Create New Product Token Form Page.

Figure 4.16 shows the Create New Product Token form page. The farmer is required to fill in all mandatory fields, including Product Name, Batch Number, Production Date, Product Quantity, Production Location, and Product Image. There is also an option to upload a certification document, if available. After clicking the Add button, a MetaMask pop-up window will appear, prompting the user to confirm the transaction request. Once confirmed, the product token will be created, and the transaction record will be stored on the blockchain.

4.2.2.6 View Product Token Page

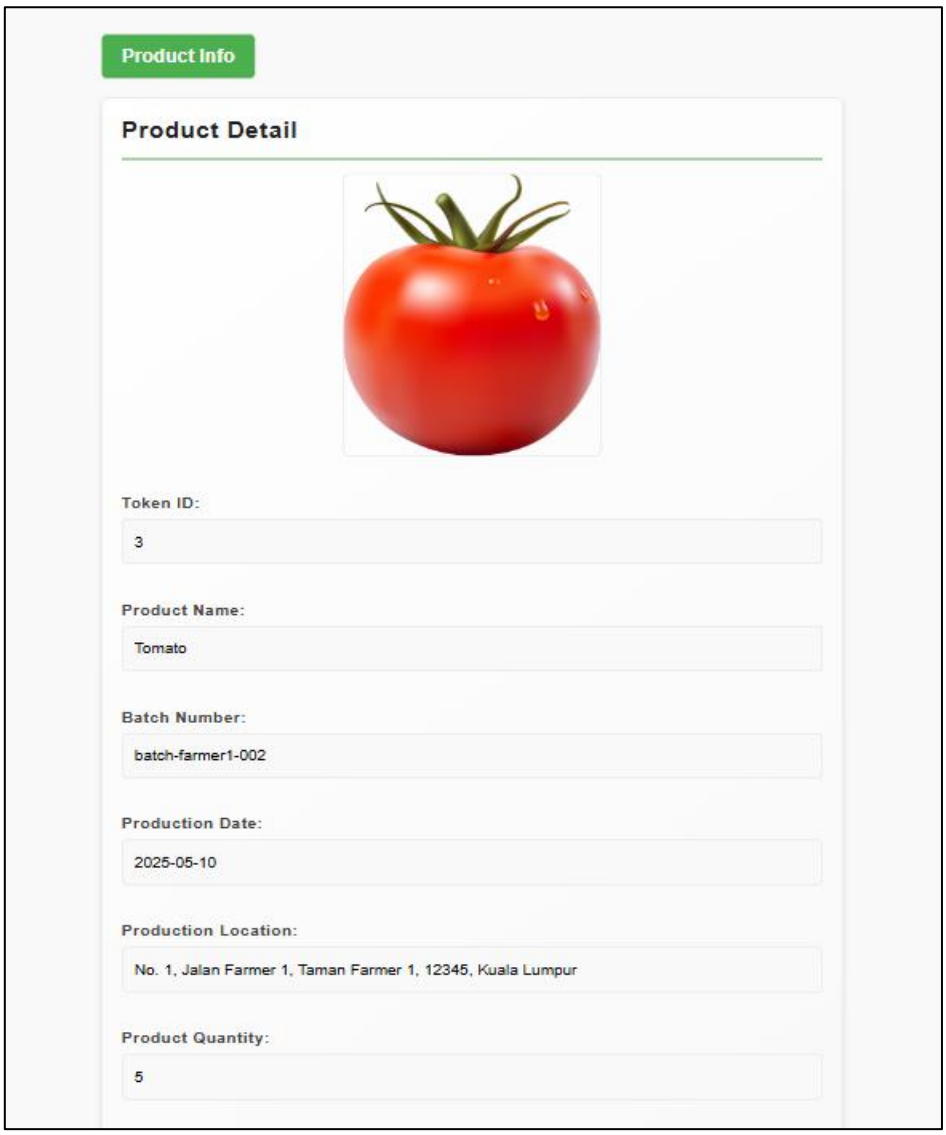


Figure 4.17: The Product Details of the Token

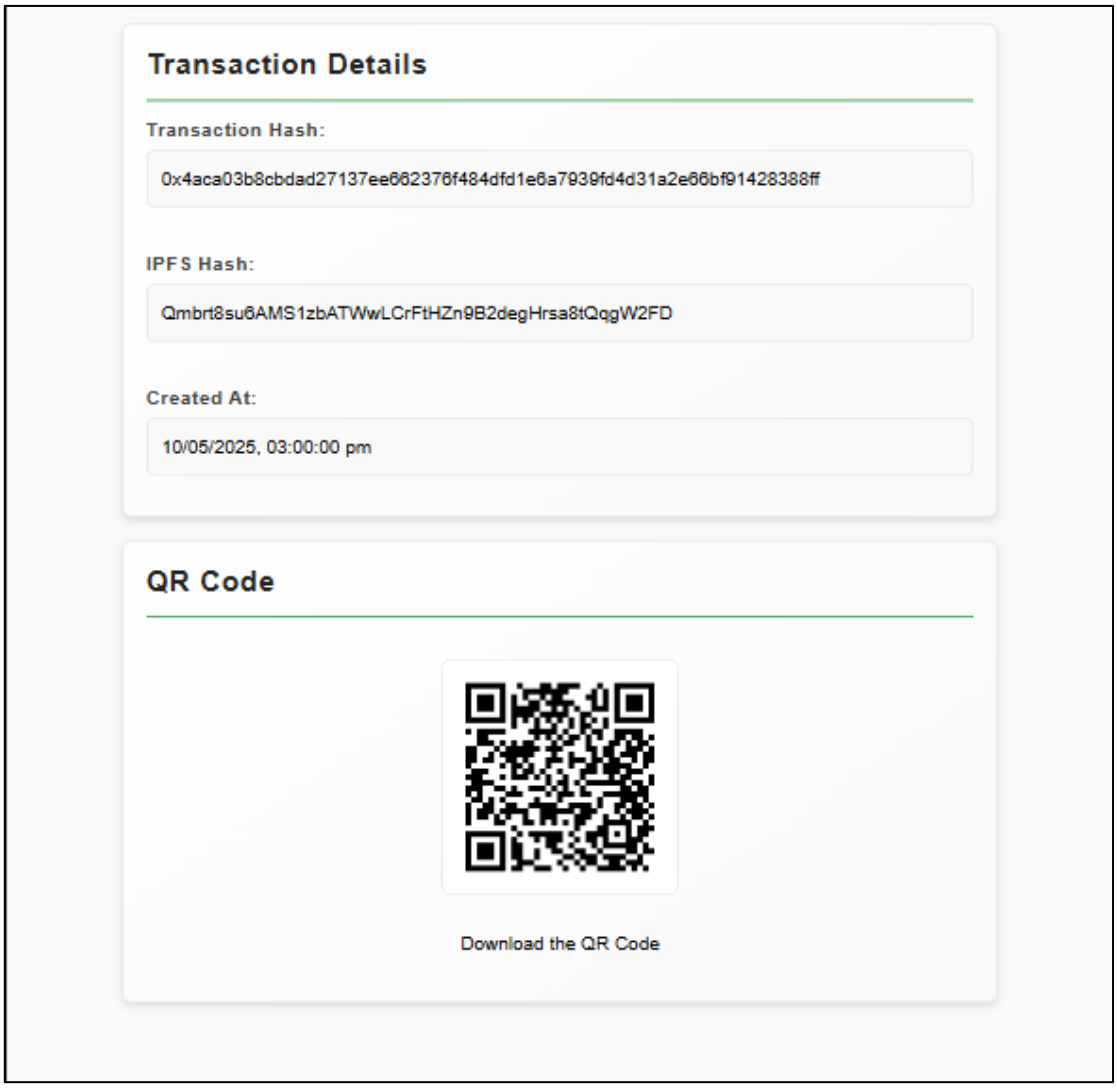


Figure 4.18: The Transaction Details of the Token

The View Product Token page consists of several sections, including Product Details and Transaction Details. As shown in Figure 4.17, the Product Details section displays information such as the Token ID, Product Name, Batch Number, Production Date, Production Location, and Product Quantity.

The Transaction Details section, shown in Figure 4.18, includes the Transaction Hash, the IPFS Data Hash, and the Timestamp indicating when the transaction was completed. At the bottom of the page, a QR code is generated for the product token. Users can scan this QR code to track the product's data in the supply chain.

4.2.2.7 Transfer Ownership of Product Token

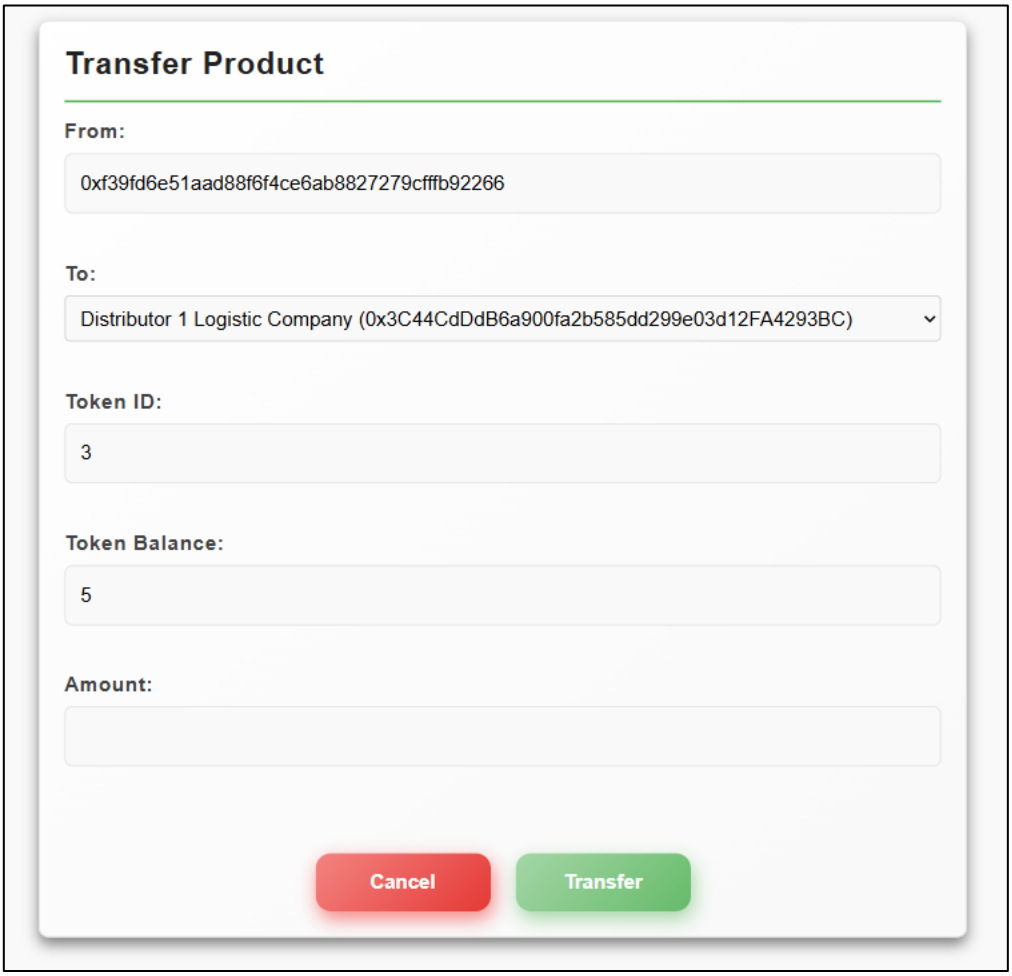


Figure 4.19: The Transfer Ownership of Product Token Form Page

Figure 4.19 shows the Transfer Ownership of Product Token form page. The From field displays the current logged-in account address. The To field shows the next party in the supply chain (e.g., Farmer → Distributor, Distributor → Retailer).

The user is required to fill in the Amount field, which must not exceed the current token balance. The token balance is displayed and will automatically update based on the amount entered. After clicking the Transfer button, a MetaMask pop-up window will appear, prompting the user to confirm the transaction request. Once confirmed, the ownership transfer record will be stored on the blockchain.

4.2.2.8 Transferred Product Token

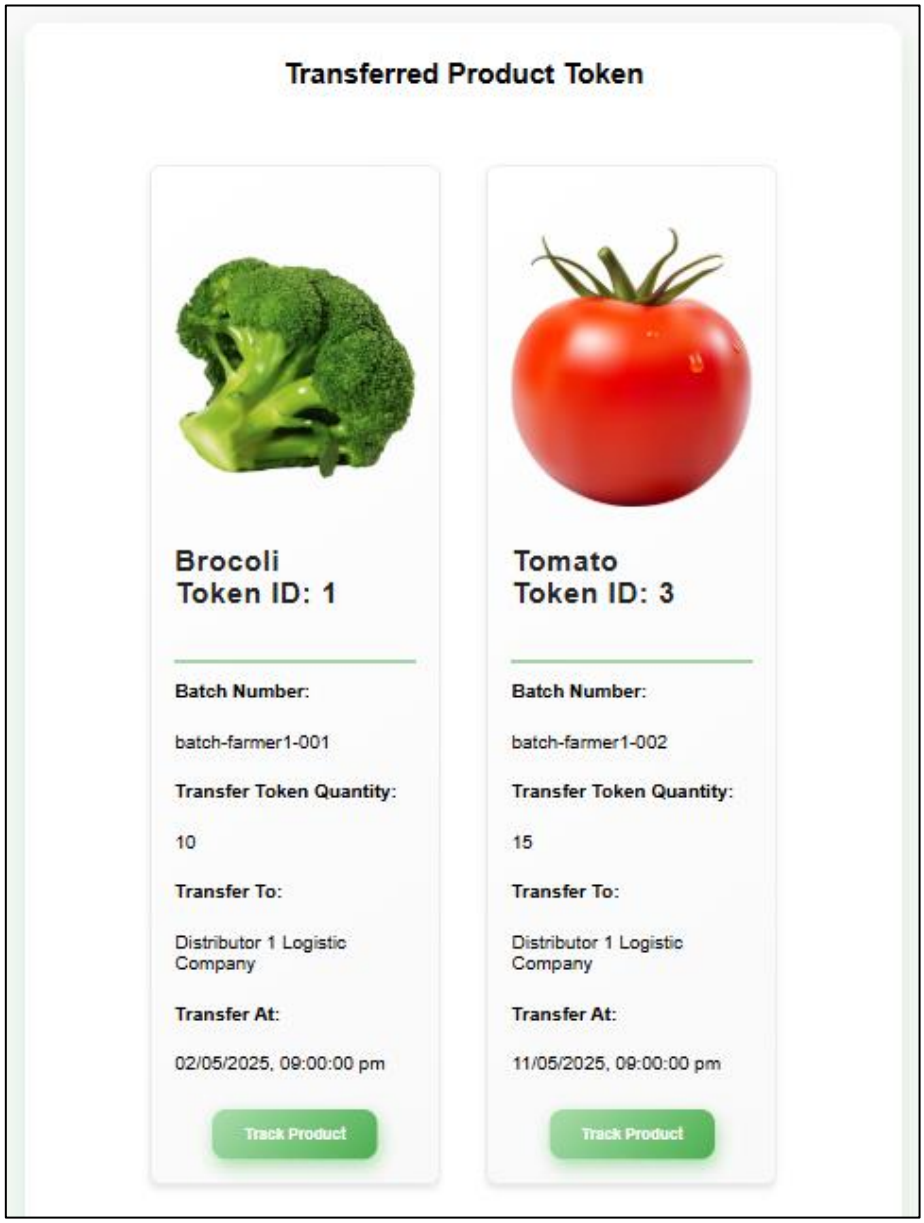


Figure 4.20: The Transferred Product Token Page

Figure 4.20 shows the Transferred Product Token page. Each product is displayed in a card format, showing essential details such as the Product Name, Token ID, Batch Number, Transferred Quantity, Transferred To, and the Date of Transfer. Each card includes a Track Product button, which allows the user to view the product's journey and status in the supply chain.3]

4.2.2.9 Track Full History Information of Product Token

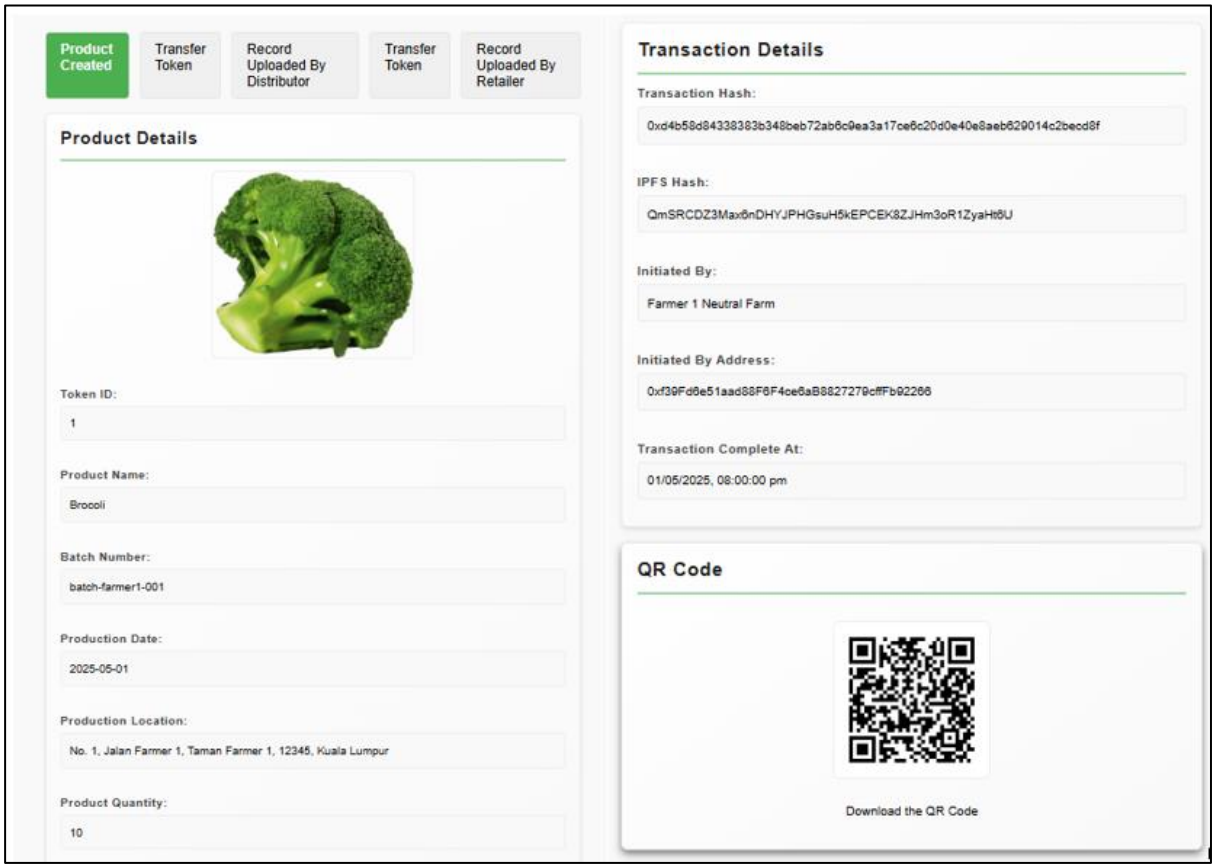


Figure 4.21: The Track Full History Info of Product Token

Figure 4.21 shows the Track Full History Information page of a product token. The tabs at the top represent the different stages of the product token in the supply chain. When the user clicks on a tab, the system displays the corresponding data for that phase. Each stage includes transaction details to ensure the traceability and authenticity of the product throughout its journey.

4.2.2.10 Add Delivery Data of Product Token

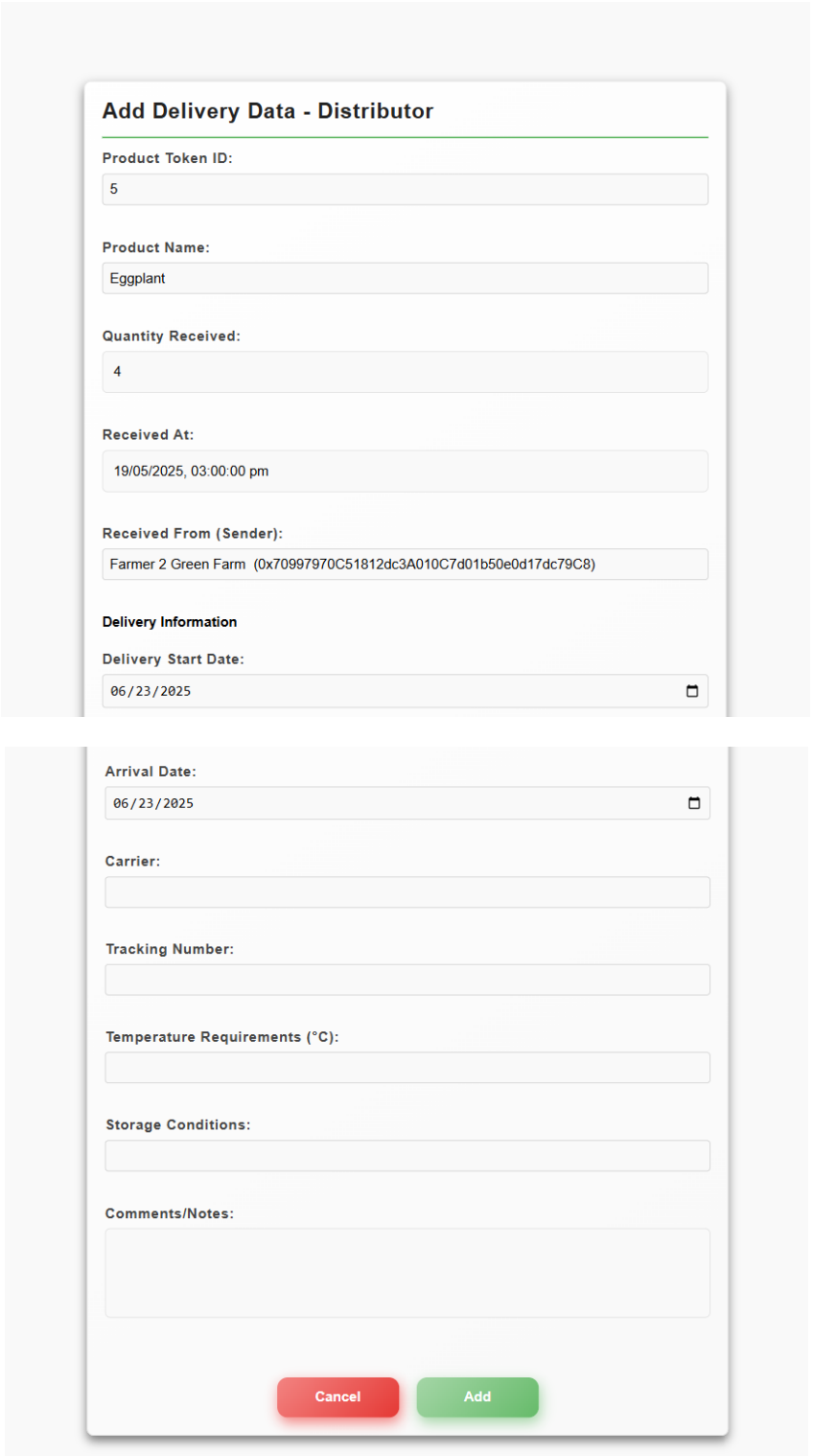


Figure 4.22: The Add Delivery Data of Product Token Form Page

Figure 4.22 shows the Add Delivery Data form page for a product token. The product information—such as Token ID, Product Name, Quantity Received, Received Time, and Received From—is displayed at the top of the form. The user is required to fill in the delivery information, including the Delivery Start Date and Arrival Date. The Comment field is optional. The system ensures that the Delivery Start Date cannot be earlier than the Received Time, and the Arrival Date must be later than the Delivery Start Date. After filling in the required fields, the user can click the Add button. A MetaMask pop-up window will appear, prompting the user to confirm the transaction request.

4.2.2.11 Add Product Data- Retailer

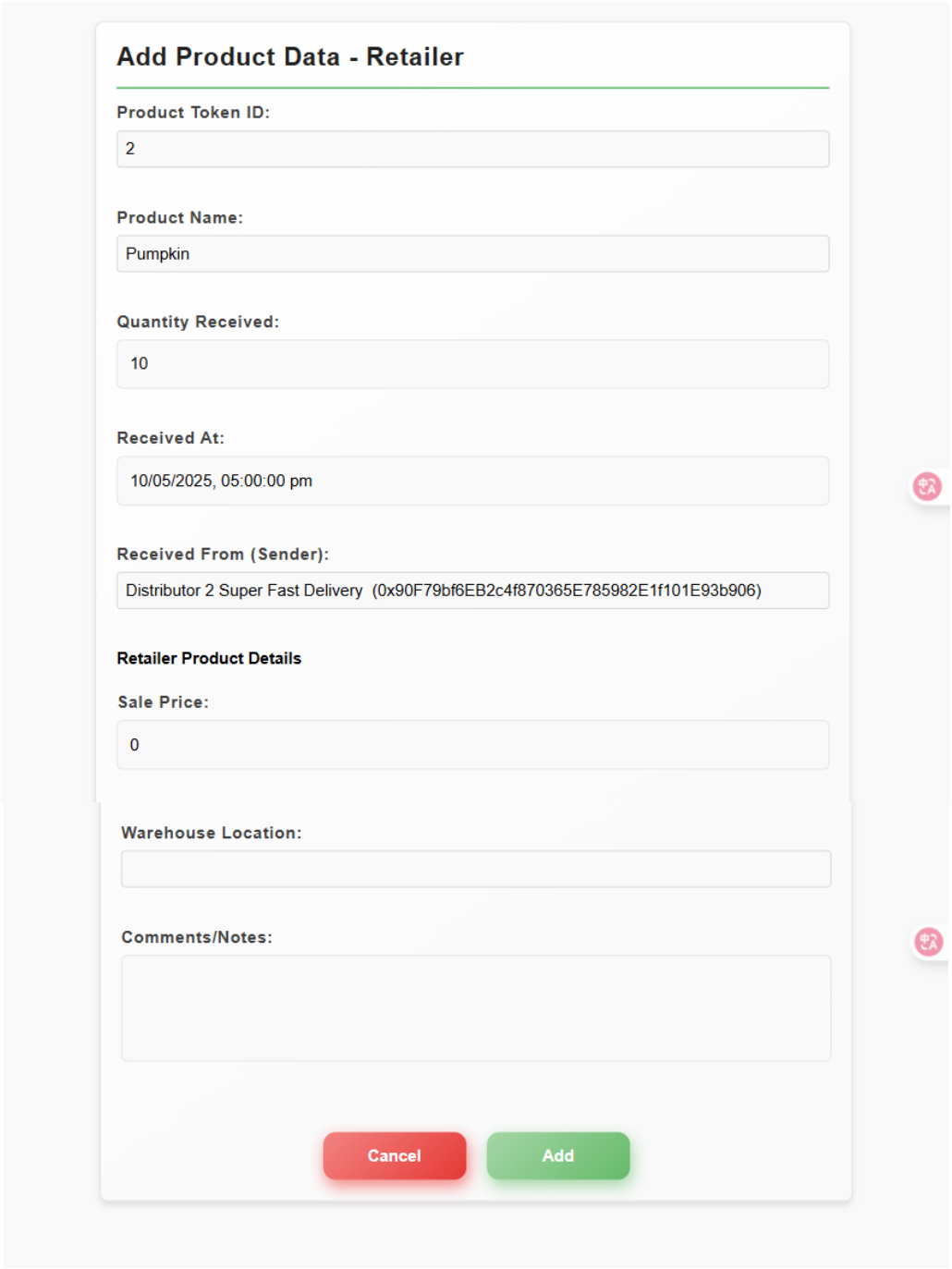


Figure 4.23: Add Product Data – Retailer Page.

Figure 4.23 shows the Add Product Data form page for the Retailer. The product information—such as Token ID, Product Name, Quantity Received, Received Time, and Received From—is displayed at the top of the form. The user is required to fill in the retailer

product details, including the Sale Price, Warehouse Location, and Comments. After filling in the required fields, the user can click the Add button. A MetaMask pop-up window will appear, prompting the user to confirm the transaction request.

4.2.2.12 Profile Detail

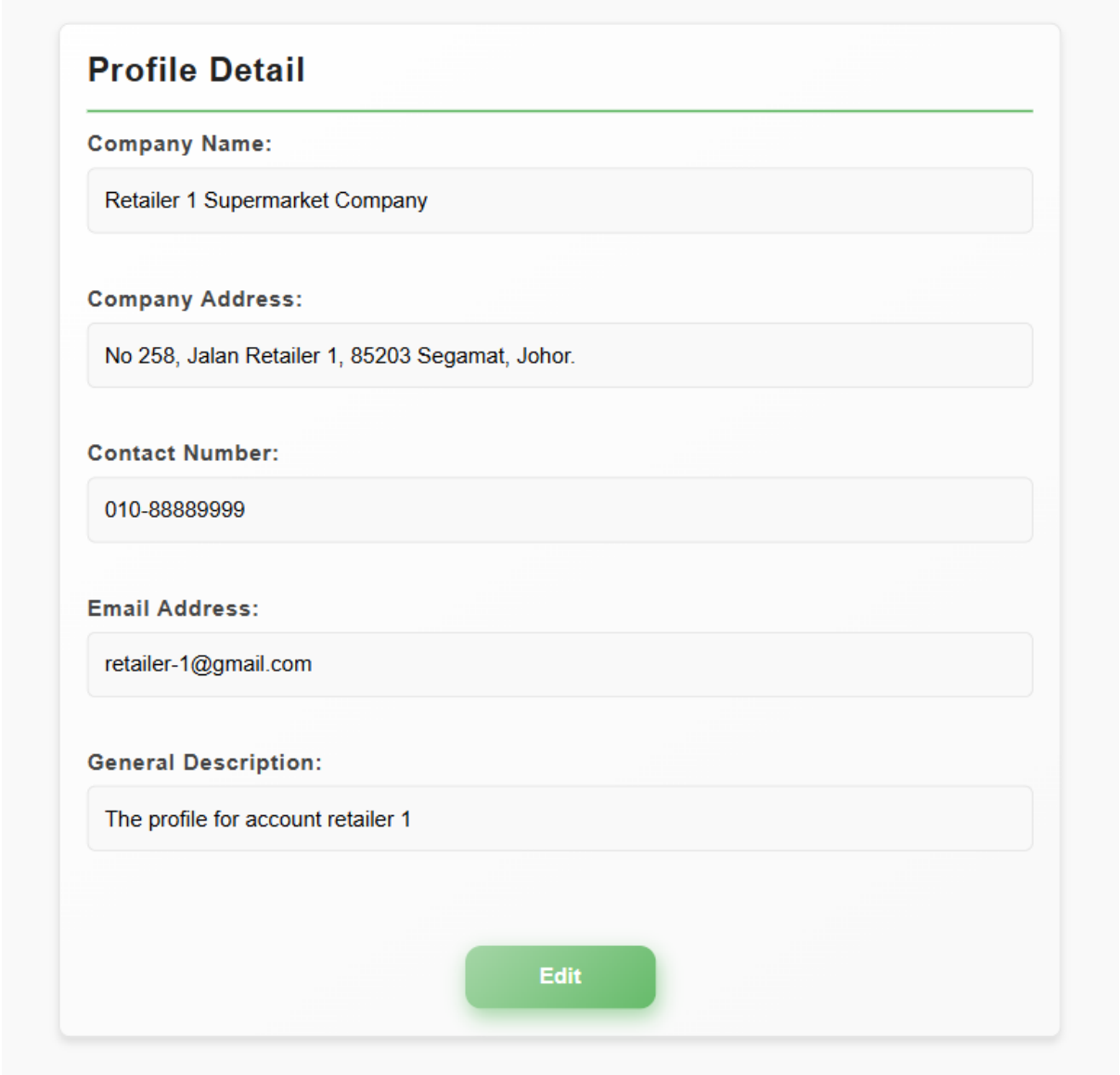


Figure 4.24: Profile Detail Page

Figure 4.24 shows the Profile Detail page. It displays the user's information, including Company Name, Company Address, Contact Number, Email Address, and a General Description. The user can click the Edit button to enter edit mode and update their profile details.

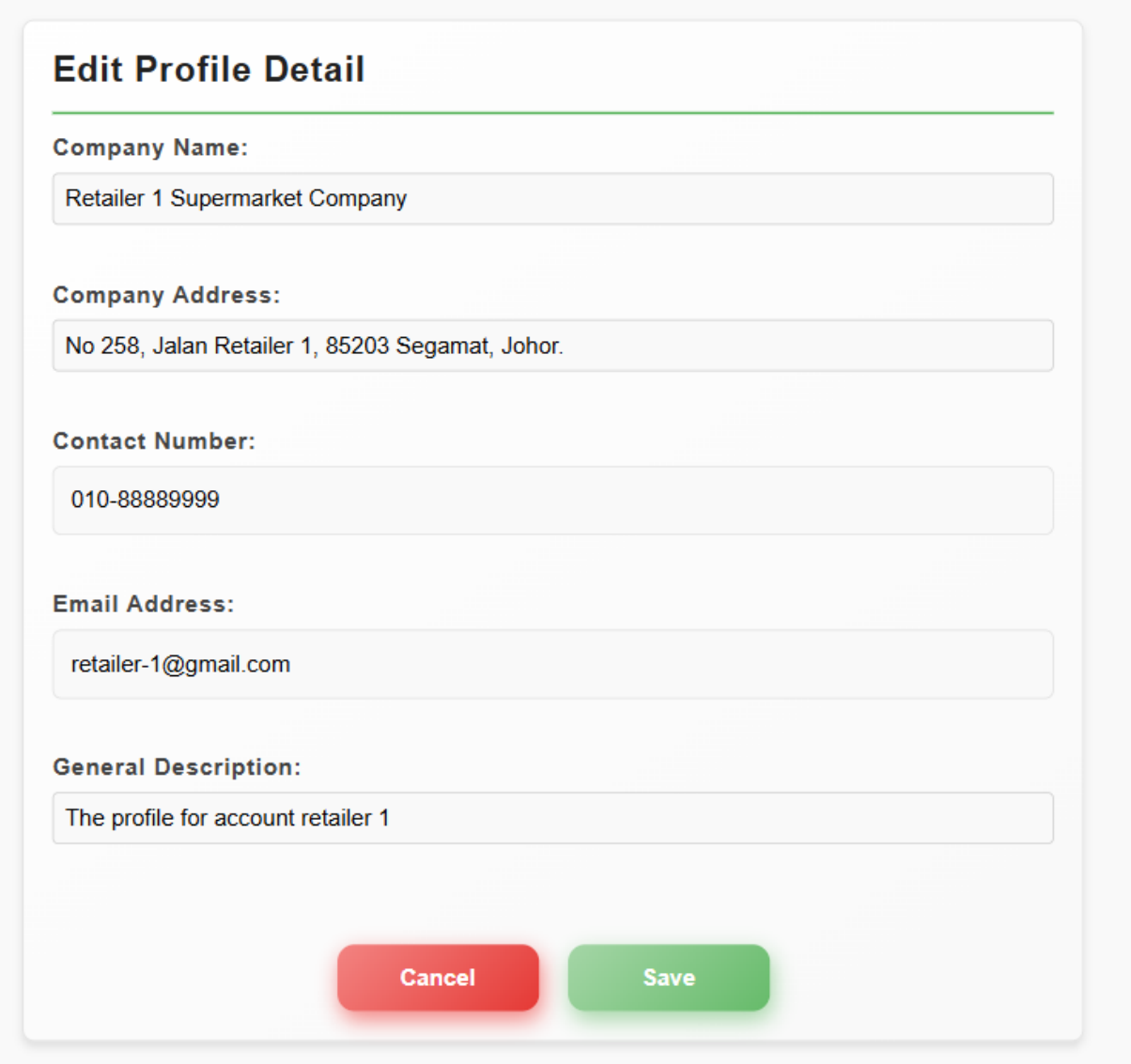


Figure 4.25: The Edit Profile Form Page

Figure 4.25 shows the Edit Profile form page. Each field is pre-filled with the user's existing profile data as the default values. After editing the desired fields, the user can click the Save button. A MetaMask pop-up window will appear, prompting the user to confirm the transaction request.

4.2.2.13 Registration Request

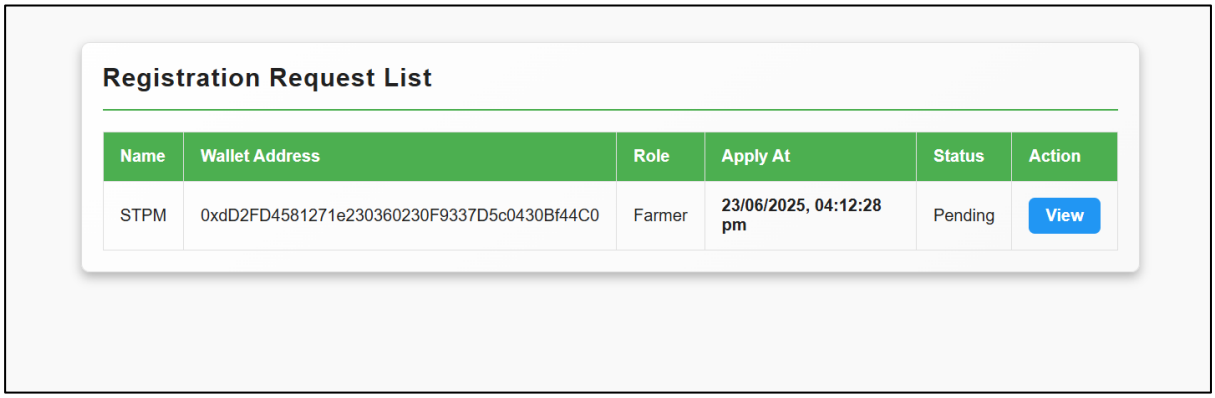


Figure 4.26: Registration Request List

Figure 4.26 shows the list of registration requests submitted by users. Each entry in the table displays relevant details of the request, including the user's Name, Wallet Address, Role, and the Timestamp when the request was made. The Status of each request is also shown, indicating whether the request is pending, approved, or rejected. The admin has the option to click the View button next to each request to access more detailed information regarding the specific request.

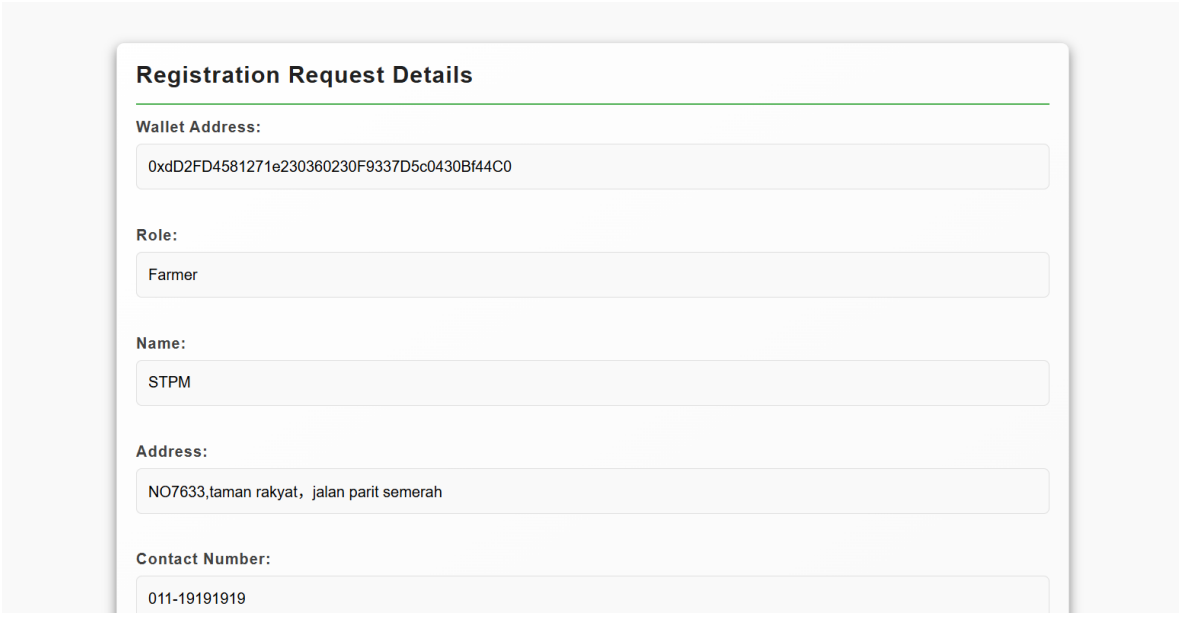


Figure 4.27: The Registration Request Details (A).

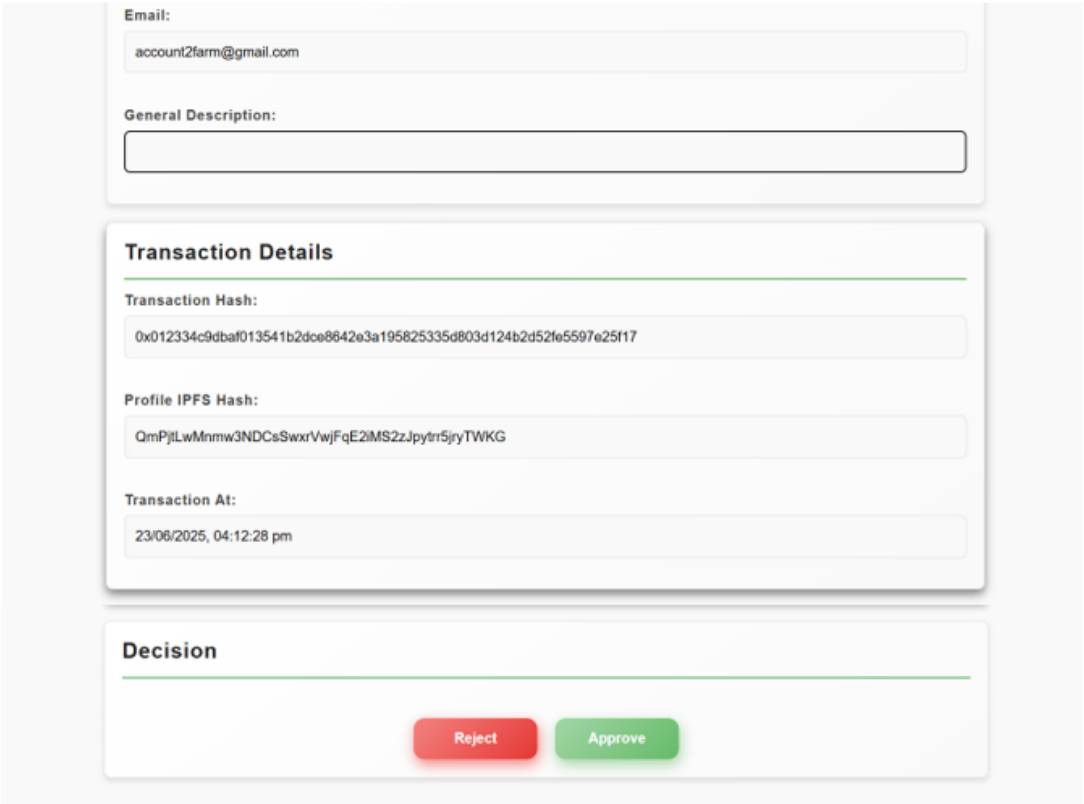


Figure 4.28: The Registration Request Details (B).

Figure 4.29: Modal Box to Fill Rejection Reason.

Figure 4.27 and

Figure 4.28: The Registration Request Details (B).

display the detailed view of a registration request. The Registration Request Details section presents essential information such as the Applier's Wallet Address, Role, Name, Address, Contact Number, Email, and a General Description provided by the applier. The Transaction Details section shows the Transaction Hash, Profile IPFS Hash, and the Transaction Occurrence Time, providing key transaction-related information. The admin has the option to either Approve or Reject the request. If the Reject option is selected, a modal box will pop up as shown as Figure 4.29, prompting the admin to provide a reason for the rejection. After the admin approves the request, a MetaMask pop-up window will appear, asking the user to confirm the transaction request. The user must approve the transaction for it to proceed.

4.2.2.14 User List

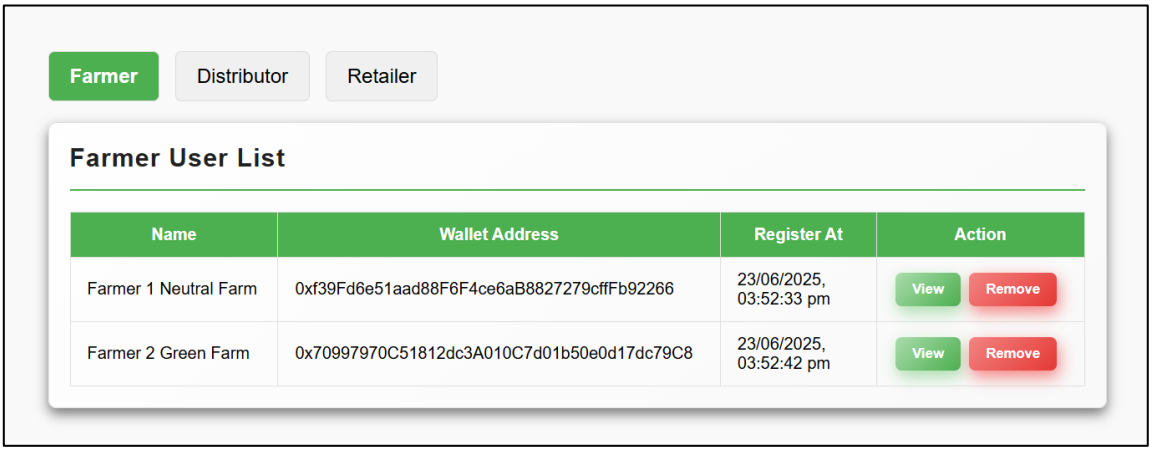


Figure 4.30: User List

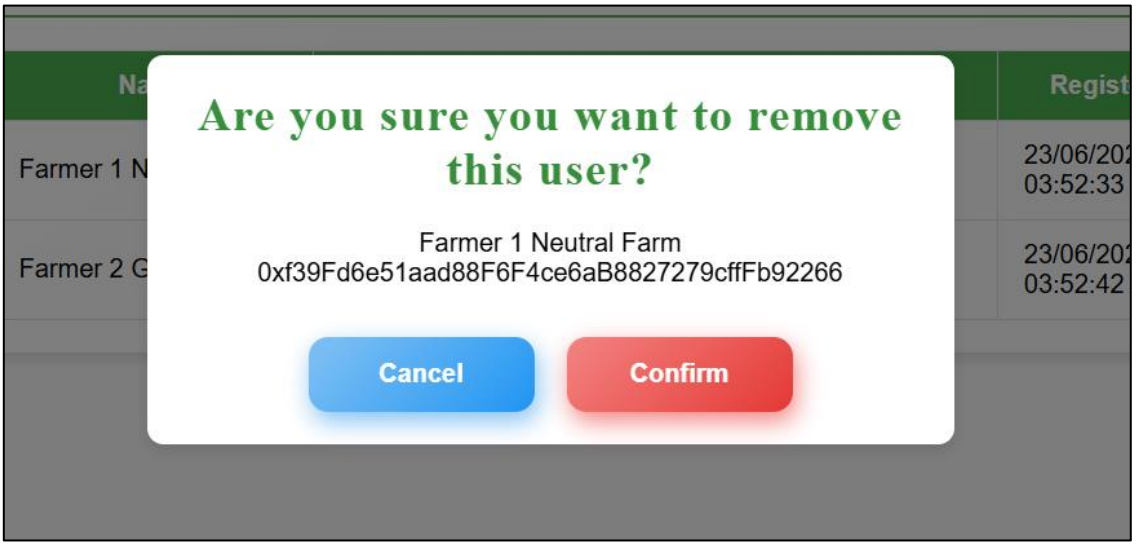


Figure 4.31: Confirmation Modal Box.

Figure 4.30 shows the user list displayed in a table format. The tabs above the table represent different user roles. The admin can click on a specific tab to view the list of users corresponding to that role. The table presents key information for each user, including the username, wallet address, and the account registration timestamp. The admin has the option to click the View button to access the detailed profile of a selected user.

Additionally, the admin can click the Remove button to delete a user account. Upon clicking the Remove button, a Confirmation Modal Box will appear as shown as Figure 4.31,

prompting the admin to confirm the action. After confirming, a MetaMask pop-up window will appear, asking the admin to approve the transaction request for account removal.

4.3 System Testing

This section focuses on the system testing phase, which is crucial for validating the functionality and performance of the proposed system. System testing plays a vital role in the software development process, ensuring that the system operates as expected under real-world conditions. This chapter will outline the various tests conducted to ensure that the system meets its design requirements and provides a stable, reliable experience for end users. The following sections will detail the implementation and results of the system testing process.

4.3.1 Functional Testing

Functional testing is a type of testing that ensures the system's functionalities work as intended, validating that each feature performs according to the defined requirements. The core modules for functional testing are the product module, profile module, and admin module. The test cases for each module are shown below.

Table 4.1: Test Case for Login System.

Test Case ID	LOGIN_01
Test Tittle	Login System
Test Description	Verify that the system login function works correctly with MetaMask authentication.
Pre-Requisite	• The user must be registered in the system. • The user must have a MetaMask account.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. From the main page, click the login button. 2. The login page should be displayed. 3. Click the login button on the page. 4. The MetaMask window will pop up, requesting the user to sign the message for authentication. 5. Click the Confirm button in the MetaMask window to authorize the login. 6. After the user confirms, the login should be successful, and the user will be logged into the system.	Account Address: 0x15d34AAf5426 7DB7D7c367839 AAf71A00a2C6A65	• The system should successfully authenticate the user via MetaMask and log them in. • The user should be redirected to their dashboard after successful login.	System performs as expected	Pass	-

Table 4.2: Test Case for Register Account

Test Case ID	REGISTER_01
Test Tittle	Register Account
Test Description	Verify that the user can successfully register an account using MetaMask for transaction confirmation.
Pre-Requisite	• The user must have a MetaMask account. • The account must not be registered yet.
Post-Requisite	The product should be successfully added to the system, and the respective product token should be generated and accessible.

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. From the main page, click the login button. 2. The login page should be displayed. 3. Click the register link. 4. The registration form will be displayed. 5. Fill in the form with the required details. 6. Click the register button. 7. The MetaMask window will pop up, requesting confirmation for the transaction. 8. Click the Confirm button in the MetaMask window to authorize the registration. 9. A successful message should be displayed confirming the account registration.	Role: Farmer Name: Test Farm Address: No 9, Jalan Testing. Contact Number: 011-2323-5858 Email Address: testFarm123@gmail.com General Description: Testing Farm Description	• The system should successfully register the user account and the user's details should be stored. • A success message should appear, confirming the account has been registered.	System performs as expected	Pass	-

Table 4.3: Test Case for Create Product Token.

Test Case ID	PRODUCT_ADD_01
Test Tittle	Create Product Token
Test Description	Verify that the farmer can successfully add a product in the system. After the product is added, the system should generate a product token.
Pre-Requisite	The farmer must be logged into the system.
Post-Requisite	The product should be successfully added to the system, and the respective product token should be generated and accessible.

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
10. Login to the system. 11. Navigate to the "Add Product" section from the sidebar. 12. Enter the required product details. 13. Click the "Add" button. 14. The MetaMask window will pop up for transaction confirmation. 15. Click the "Confirm" button in the MetaMask window. 16. Redirect to the created product info page with a success message	Product Name: Banana Batch Number: Batch_Farmer1_020 Production Date: 06/15/2025 Product Quantity: 50 Production Location: No1, Jalan Kuching 20 Product Image: Upload image file	• The product token is created successfully with the correct details. • The blockchain transaction related to the token creation is recorded accurately on the network. • The system redirects the user to the correct product token info page	System performs as expected	Pass	-

Table 4.4: Test Case for View Owned Product List.

Test Case ID	PRODUCT _VIEW_01
Test Tittle	View Owned Product List
Test Description	Verify that the system correctly displays the user's owned product tokens.
Pre-Requisite	• The user is logged into the system. • The user has at least one owned product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Login to the system. 2. Navigate to the "Owned Product Tokens" section from the sidebar. 3. Verify that the owned product tokens are displayed correctly, with all relevant details such as token ID, token name and token quantity.	N/A	• The system correctly displays the list of owned product tokens associated with the user. • The product token information should be accurate and up to date.	System performs as expected	Pass	-

Table 4.5: Test Case for View Transferred Product Token List.

Test Case ID	PRODUCT _VIEW_02
Test Tittle	View Transferred Product Token List
Test Description	Verify that the system correctly displays the user's transferred product tokens.
Pre-Requisite	• The user is logged into the system. • The user has at least one transferred product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Login to the system. 2. Navigate to the "Transferred Product Tokens" section from the sidebar. 3. Verify that the transferred product tokens are displayed correctly, with all relevant details such as token ID, token name and token quantity.	N/A	• The system correctly displays the list of transferred product tokens, showing accurate details for each token. • The product token information should be accurate and up to date.	System performs as expected	Pass	-

Table 4.6: Test Case for View Full Complete Info of the Product Token.

Test Case ID	PRODUCT _SHOW_01
Test Tittle	View the Full Complete Info of the Product Token
Test Description	Verify that the system correctly displays the full details of the product token.
Pre-Requisite	• The user is logged into the system. • The user has at least one product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid credentials. 2. Navigate to the "Owned Product Tokens" section from the sidebar. 3. Select a product token from the list and click the view detail button. 4. Verify that the system displays the complete product token information correctly, including: • Product Info (e.g., name, batch number, etc) • Creator address • Timestamp of creation • Transaction hash • IPFS CID	N/A	The system correctly displays the full details of the selected product token, with all relevant information such as the creator address, timestamp, and transaction hash.	System performs as expected	Pass	-

Table 4.7; Test Case for Track Full Timeline of Historical Info of the Product Token.

Test Case ID	PRODUCT _SHOW_02
Test Tittle	Track the Full Timeline of Historical Info of the Product Token
Test Description	Verify that the system correctly tracks and displays the full timeline of historical details of the product token.
Pre-Requisite	• The user is logged into the system. • The user has at least one product token. • The product token has multiple historical data records.
Post-Requisite	-

Test Steps	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Comments**
1. Log in to the system using valid credentials. 2. Navigate to the " Transferred Product Tokens" section from the sidebar. 3. Select a product token from the list and click the track button. 4. Verify that the system displays the complete historical information of the product token correctly, including: • The name of the stage/phase (e.g., created, transferred, verified, etc.) • Transaction hash • Address involved in the transaction • Timestamp of the action • IPFS CID for new added data	N/A	• The system correctly displays the full historical timeline of the selected product token. • All relevant information is shown for each phase, such as the name of the stage, transaction hash, involved addresses, timestamp, and any additional data.	System performs as expected	Pass	-

Table 4.8: Test Case for Transfer the Product Token.

Test Case ID	PRODUCT _TRANSFER_01
Test Tittle	Transfer the Product Token.
Test Description	Verify that the system correctly transfers the product token to another user, and the relevant transaction record should be stored on the blockchain.
Pre-Requisite	• The user is logged into the system. • The user has at least one product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid credentials. 2. Navigate to the "Owned Product Tokens" section from the sidebar. 3. Select a product token from the list and click the "Transfer" button. 4. Fill in the transfer input based on test data. The dropdown list for selecting the recipient should display options based on the user role: • If the user is a farmer, only distributors should be shown as options.	Amount: Use the token balance shown for the selected product token to fill in the amount field.	• The system successfully transfers the product token to another user. • The transaction information should be correctly recorded and stored on the blockchain. • The recipient should receive the transferred product token.	System performs as expected	Pass	-

• If the user is a distributor, only retailers should be shown as options. 5. Select the recipient and click the "Transfer" button. 6. The MetaMask window will pop up for transaction confirmation. 7. Click the "Confirm" button in the MetaMask window. 8. A "Transfer Successful" message should be displayed.					

Table 4.9: Test Case for Update the Delivery Info of Product Token.

Test Case ID	PRODUCT _UPDATE_01
Test Tittle	Update the Delivery Info of Product Token
Test Description	Verify that the system correctly updates the delivery info of the product token, and the relevant updated record should be stored on the blockchain.
Pre-Requisite	• The distributor is logged into the system. • The distributor has at least one product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid credentials. 2. Navigate to the "Owned Product Tokens" section from the sidebar. 3. Select a product token from the list and click the "Add Delivery Info" button. 4. The system should display the Add Delivery Data page. 5. The system should show the correct product info, including the product token ID, name, received at, and received from.	Delivery Date: current date Arrival Date: 2 days after current date Carrier: Thomas Tracking Number: MY167589 Temperature Requirements: 10	• The system successfully updates the delivery information for the product token. • The updated data event should be correctly recorded and stored on the blockchain. • The updated information should be displayed and saved correctly in the system.	System performs as expected	Pass	-

6. Fill in the input fields based on the provided test data. 7. Click the "Add" button to submit the updated information. 8. The MetaMask window will pop up for transaction confirmation. 9. Click the "Confirm" button in the MetaMask window. 10. A successful message should be displayed.	Storage Conditions: In cool and dry Comments/Notes: Please delivery carefully.				

Table 4.10: Test Case for Update the Retailer Info of Product Token

Test Case ID	PRODUCT _UPDATE_02
Test Tittle	Update the Retailer Info of Product Token
Test Description	Verify that the system correctly updates the retailer info of the product token, and the relevant updated record should be stored on the blockchain.
Pre-Requisite	• The retailer is logged into the system. • The retailer has at least one product token.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid credentials. 2. Navigate to the "Owned Product Tokens" section from the sidebar. 3. Select a product token from the list and click the "Add Retailer Info" button. 4. The system should display the Add Retailer Data page. 5. The system should show the correct product info, including the product token ID, name, received at, and received from. 6. Fill in the input fields based on the provided test data.	Sales Price: 12.50 Warehouse Location: Batu Gawa Kuching Comments/Notes: Store in cool dry places.	• The system successfully updates the retailer information for the product token. • The updated data event should be correctly recorded and stored on the blockchain. • The updated information should be displayed and saved correctly in the system.	System performs as expected	Pass	-

7. Click the "Add" button to submit the updated information. 8. The MetaMask window will pop up for transaction confirmation. 9. Click the "Confirm" button in the MetaMask window. 10. A successful message should be displayed.					

Table 4.11: Test Case for Scan QR Code to View Full Historical Data of Product Token

Test Case ID	PRODUCT _TRACK_01
Test Tittle	Scan QR Code to View Full Historical Data of Product Token
Test Description	Verify that the system can scan the product QR code and show the full historical data and info of the product token.
Pre-Requisite	The product token should have a completed historical information.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Click the "Consumer Traceability" button on the main page. 2. The system will display the "Trace Product Information" page. 3. Click the "Start Scanning" button. The camera interface will appear. 4. Use the camera to scan the QR code of the product token. 5. The system should display the full historical information of the product token.	N/A	• The system successfully scans the QR code and redirects to the product token information page. • The system successfully displays the historical information for the product token.	System performs as expected	Pass	-

Table 4.12: Test Case for View User Profile.

Test Case ID	PROFILE_VIEW_01
Test Tittle	View User Profile
Test Description	Verify that the system correctly displays the profile data of the user.
Pre-Requisite	The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Comments**
1. Log in to the system using valid credentials. 2. Navigate to the "Profile" section from the sidebar. 3. Verify that the profile data is correctly displayed, including user details such as name, email, contact number, and any other relevant information stored in the profile.	-.	The system should successfully show the profile data of the user, such as name, email address, and contact number.	System performs as expected	Pass	-

Table 4.13: Test Case for Edit User Profile.

Test Case ID	PROFILE_EDIT_01
Test Tittle	Edit User Profile
Test Description	Verify that the system correctly updates the profile data of the user.
Pre-Requisite	The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid credentials (e.g., username/email and password). 2. Navigate to the "Profile" section from the sidebar. 3. Click the "Edit" button to allow changes to the profile. 4. Fill in the profile fields with the new data: 5. Click the "Confirm" button to save the changes. 6. The MetaMask window will pop up for transaction confirmation. 7. Click the "Confirm" button in the MetaMask window to authorize the transaction.	Company Name: Testing New Name Company Address: No188, Jalan Testing. Contact Number: 011-63639898 Email Address: testingData@gmail.com General Description: The testing description data.	The system should successfully update the profile data of the user, and a confirmation message should appear to indicate the changes were successfully applied.	System performs as expected	Pass	-

8. A successful message should be displayed confirming the profile update.					

Table 4.14: Test Case for View Registration Request List.

Test Case ID	ADMIN_REG_01
Test Tittle	View Registration Request List
Test Description	Verify that the system correctly displays the registration request list.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "Registration Request" section from the sidebar. 3. Verify that the system displays the registration request list. Ensure that each registration request shows the name of the registrant, wallet address, role, and timestamp of request.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	The system should successfully display the registration request list, and for each request, it should show the name of the registrant, the wallet address, the role assigned, and the timestamp of when the request was submitted.	System performs as expected	Pass	-

Table 4.15: Test Case for View Full Registration Request Information.

Test Case ID	ADMIN_REG_02
Test Tittle	View Full Registration Request Information
Test Description	Verify that the admin can view the full detailed information of a registration request successfully.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "Registration Request" section from the sidebar. 3. Select a registration request from the list and click the view button. 4. The system should display the full details of the registration request, including the applier's wallet address, role, name, transaction hash, IPFS hash, and the timestamp of the transaction.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	The system should successfully display the full registration request information, including all the listed details (wallet address, role, name, transaction hash, IPFS hash, and timestamp) in a clear and organized manner.	System performs as expected	Pass	-

Table 4.16: Test Case for Approve The Registration Request.

Test Case ID	ADMIN_REG_03
Test Tittle	Approve The Registration Request
Test Description	Verify that the admin can approve the registration request successfully.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "Registration Request" section from the sidebar. 3. Select a registration request from the list and click the view button. 4. Review the registration details. 5. Click the approve button to approve the registration. 6. The MetaMask window will pop up for transaction confirmation. 7. Click the Confirm button in the MetaMask window to authorize the transaction. 8. A successful message should be displayed, confirming the approval of the registration request.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	• The system should successfully approve the registration request. • A success message should appear, confirming that the registration request has been approved.	System performs as expected	Pass	-

Table 4.17: Test Case for Reject The Registration Request.

Test Case ID	ADMIN_REG_04
Test Tittle	Reject The Registration Request
Test Description	Verify that the admin can successfully reject a registration request.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "Registration Request" section from the sidebar. 3. Select a registration request from the list and click the view button. 4. Review the registration details. 5. Click the reject button to reject the registration request. 6. A modal box will appear, asking the admin to input the reason for rejection. 7. Input the reason for rejection and confirm. 8. The MetaMask window will pop up for transaction confirmation. 9. Click the Confirm button in the MetaMask window to authorize the transaction.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9 Reason Rejection: Scammer Account	• The system should successfully approve the registration request. • A success message should appear, confirming that the registration request has been approved.	System performs as expected	Pass	-

10. A successful message should be displayed, confirming the rejection of the registration request.					

Table 4.18: Test Case for View and Filter User List Based on Role.

Test Case ID	ADMIN_USER_01
Test Tittle	View and Filter User List Based on Role
Test Description	Verify that the admin can view and filter the user list based on different roles, and ensure the correct user information is displayed.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "User List" section from the sidebar. 3. Select a role from the available role filter tab (e.g., Farmer, Distributor, Retailer) 4. The system should update the user list to reflect only the users with the selected role. 5. Verify that for each user listed, the system displays the correct and complete information, including the name, wallet address, and timestamp of registration.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	• The system should filter and display users based on the selected role, ensuring that only users with the selected role are shown in the list. • The system should display the correct information for each user, which should include their name, wallet address, and	System performs as expected	Pass	-

		the timestamp of their registration.			

Table 4.19: Test Case for View the Full Details of the User.

Test Case ID	ADMIN_USER_02
Test Tittle	View the Full Details of the User
Test Description	Verify that the admin can view the complete details of a user, including personal and account information.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "User List" section from the sidebar. 3. Select a user from the list and click the view button. 4. Verify that the system displays the full details of the selected user, including their wallet address, name, contact number, and address.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	• The system should display the complete information for the selected user, including their wallet address, name, contact number, and address, all clearly shown in the user details section.	System performs as expected	Pass	-

Table 4.20: Test Case for Remove the User from the User List.

Test Case ID	ADMIN_USER_03
Test Tittle	Remove the User from the User List
Test Description	Verify that the admin can successfully remove a user from the user list.
Pre-Requisite	• The user must have admin role. • The user must be logged into the system with valid credentials.
Post-Requisite	-

Test Steps	Inputs	Expected Output	Actual Output	Test Result	Comments
1. Log in to the system using valid admin credentials. 2. Navigate to the "User List" section from the sidebar. 3. Select a user from the list and click the remove button. 4. A confirmation modal box will appear, asking the user to confirm the action. 5. Click the confirm button to proceed with the removal. 6. The MetaMask window will pop up for transaction confirmation. 7. Click the Confirm button in the MetaMask window to authorize the transaction.	Login Wallet Address: 0x976EA74026E726554dB657fA54763abd0C3a0aa9	• The user should be successfully removed from the user list, and their data should no longer be displayed in the system. • A success message should appear, confirming that the user has been successfully removed from the list.	System performs as expected	Pass	-

8. A successful message should be displayed, confirming that the user has been successfully removed.					

4.3.2 Non-Functional Testing

Non-functional testing ensures that the software meets non-functional requirements, such as performance and usability. This section focuses on testing the modules' behaviour under various conditions that do not directly relate to the core functionality but are essential for a smooth user experience.

Table 4.21: Test Case for QR Code Response Time.

Test Case ID	Performance_01				
Test Tittle	Test QR Code Response Time for Product Data Retrieval				
Test Description	Verify that the average response time for retrieving product data via QR code should not exceed 3 seconds.				
Pre-Requisite	The product token must have full historical data.				
Post-Requisite	-				
Test Steps		**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Comments**
1. Navigate to the "Consumer Traceability" section. 2. The system should display the "Trace Product Information" page.		-	The average response time for retrieving product data via QR code should not exceed 3 seconds.	The average response time for retrieving product	Pass	-

3. Click the "Start Scanning" button. 4. Ensure that the camera of the device is functioning and visible on the screen. 5. Scan the QR code of the product. 6. Measure and verify that the response time for displaying the product's historical data is less than 3 seconds. 7. The system should display the full historical data of the product token.			data via QR code is 2 seconds.		

Table 4.22: Test Case for Mobile-friendly.

Test Case ID	Usability_01
Test Tittle	Mobile-Friendly Design for System Access on Smartphones
Test Description	Verify that the system's design is mobile-friendly and supports users accessing it on smartphones.
Pre-Requisite	The system should be accessible via mobile browsers.
Post-Requisite	-

Test Steps	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Comments**
1. Open the system in a mobile browser on a smartphone. 2. Ensure that the layout and design are responsive and adapt to different screen sizes. 3. Verify that the navigation menus, buttons, and text are easily readable and accessible on a small screen. 4. Ensure that interactive elements such as buttons and forms are appropriately sized and functional. 5. Test common mobile gestures (e.g., scrolling, tapping, zooming) and verify that the system behaves as expected. 6. Ensure that key features of the system are easily accessible without the need for horizontal scrolling.	-	• The system's design should be responsive, ensuring that it is mobile-friendly and easy to use on smartphones. • All navigation, interactive elements, and content should be fully accessible and functional on mobile devices.	Most pages perform well on mobile devices, but some pages, such as the "Owned Product" page and the "Transferred Product" page, need design improvements to enhance the mobile experience.	Pass	-

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

This project has successfully demonstrated the application of blockchain technology to address key challenges in the food supply chain, specifically in enhancing transparency, traceability, and consumer trust. By implementing a blockchain-based system for food product tokenization and consumer verification, the project has provided an innovative solution that improves data integrity, automates ownership transfers, and streamlines refund and compensation processes. The system utilizes Ethereum-based technologies such as Hardhat, IPFS, Solidity, and ERC1155 NFTs to create a decentralized platform that enables consumers to access verified product information in real time through QR codes.

The system has been designed with the aim of overcoming the limitations of traditional centralized systems, including data tampering and inefficiencies in managing consumer complaints. Through the successful deployment of smart contracts, product ownership can be securely verified, and refunds and compensation can be automated, reducing processing time and minimizing operational costs. Additionally, the integration of blockchain and IPFS ensures that product data remains tamper-resistant, allowing consumers to trust the information they receive regarding the origins and handling of the products they purchase.

The project has achieved its objectives of improving transparency and data security while also enhancing the efficiency of supply chain processes. The outcomes indicate that blockchain technology holds significant potential in revolutionizing consumer-facing sectors, especially in food safety and traceability, and can serve as a scalable framework for future implementations.

5.2 Achievements

1. Blockchain-based Consumer Verification: The development of a system that allows consumers to scan QR codes and access product details securely and transparently. The use of product tokenization ensures that product ownership and authenticity can be easily verified.
2. Automated Refund and Compensation: The implementation of smart contracts to automate refund and compensation processes, reducing the need for manual data verification and significantly improving the speed and accuracy of transactions.
3. Decentralized Storage and Data Integrity: The use of IPFS for decentralized file storage ensures that product information and certifications are immutable and accessible, addressing the security concerns of centralized data storage.
4. User Experience and Accessibility: A user-friendly web interface built with Web 3.0 technologies (HTML, CSS, Ether.js) that allows easy interaction with the blockchain, providing both consumers and stakeholders with seamless experience.

5.3 Limitations and Constraints

While the project has successfully met its core objectives, there are several limitations and constraints that must be acknowledged:

1. Supplier-side Data Management: While the system does involve some data input from suppliers, the scope of this data is minimal and primarily limited to simple side data such as product details and delivery information. The system does not include extensive supplier-side management or detailed production data, which means that full traceability from production to the consumer is not fully realized in this project.
2. Scalability: The system is based on a simulated food supply chain environment rather than a real-world, large-scale application. Further testing and optimization are required to ensure

the system's scalability when deployed in more complex and variable supply chain scenarios.

3. Blockchain Transaction Fees: Although the Ethereum network offers a decentralized and secure platform, transaction fees (gas fees) could become a barrier for frequent interactions, especially for smaller-scale producers or low-cost products.
4. Interoperability: While blockchain offers significant advantages in data integrity and transparency, integrating the system with other existing supply chain technologies may pose challenges due to differences in protocols, standards, and data formats.

5.4 Future Work

Despite the promising results of this project, several avenues for future development and enhancement remain:

1. Advanced Supplier-side Integration: Future work should extend the system to include comprehensive supplier-side data management, enabling full traceability from farm to fork. This would involve integrating production data, quality control, inventory management, and real-time logistical updates from all supply chain participants. By incorporating IoT devices, RFID tags, and smart sensors, the system can automate data collection and ensure seamless, real-time updates, improving transparency and product authenticity across the supply chain.
2. Scalability Enhancements: To make the system suitable for large-scale, real-world deployments, scalability improvements must be made. This could involve optimizing blockchain transaction speeds, reducing latency, and exploring alternative blockchain solutions like Layer 2 protocols or more energy-efficient consensus mechanisms.
3. Cross-chain Interoperability: Exploring interoperability with other blockchain networks (e.g., Hyperledger, Solana) could expand the system’s utility and help ensure that it can be

integrated with existing supply chain technologies and platforms, facilitating smoother collaboration across multiple sectors.

4. Advanced AI and Machine Learning Integration: Future research could integrate artificial intelligence (AI) and machine learning (ML) to predict and analyze trends in the food supply chain, such as demand forecasting or fraud detection, based on the vast amounts of data generated by the blockchain system.
5. Blockchain Privacy Enhancements: As blockchain is a public ledger, additional privacy measures, such as zk-SNARKs (zero-knowledge proofs), can be implemented to protect sensitive business data while ensuring transparency for consumers.

REFERENCES

Bhatia, S., & Albarrak, A. S. (2023). A Blockchain-Driven Food Supply Chain Management Using QR Code and XAI-Faster RCNN Architecture. Sustainability, 15(3), 2579. https://doi.org/10.3390/su15032579

Bodkhe, U., Tanwar, S., Parekh, K., Khanpara, P., Tyagi, S., Kumar, N., & Alazab, M. (2020). Blockchain for Industry 4.0: A comprehensive review. *IEEE Access*, *8*, 79764–79800. https://doi.org/10.1109/access.2020.2988579

Caro, M. P., Ali, M. S., Vecchio, M., & Giaffreda, R. (2018). Blockchain-based traceability in agri-food supply chain management: A practical implementation. 2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany), Tuscany, Italy, 1–4. https://doi.org/10.1109/IOT-TUSCANY.2018.8373021

Ellahi, R. M., Wood, L. C., & Bekhit, A. E. -D. A. (2024). Blockchain-Driven Food Supply Chains: A Systematic Review for Unexplored Opportunities. *Applied Sciences*, *14*(19), 8944. https://doi.org/10.3390/app14198944

Fiore, M., & Mongiello, M. (2023). History of blockchain technology and its impact in social good. *2023 8th IEEE History of Electrotechnology Conference (HISTELCON)*, 36–38. https://doi.org/10.1109/histelcon56357.2023.10365852

Galvez, J. F., Mejuto, J., & Simal-Gandara, J. (2018). Future challenges on the use of blockchain for food traceability analysis. *TrAC Trends in Analytical Chemistry*, *107*, 222–232. https://doi.org/10.1016/j.trac.2018.08.011

Iftekhar, A., Cui, X., Hassan, M., & Afzal, W. (2020). Application of blockchain and Internet of Things to ensure tamper-proof data availability for food safety. *Journal of Food Quality*, *2020*, 1–14. https://doi.org/10.1155/2020/5385207

Javaid, M., Haleem, A., Singh, R. P., Khan, S., & Suman, R. (2021). Blockchain technology applications for Industry 4.0: A literature-based review. *Blockchain Research and Applications*, *2*(4), 100027. https://doi.org/10.1016/j.bcra.2021.100027

Kshetri, N. (2017). 1 Blockchain's roles in meeting key supply chain management objectives. *International Journal of Information Management*, *39*, 80–89. https://doi.org/10.1016/j.ijinfomgt.2017.12.005

Piccardo, G., Conti, L., & Martino, A. (2024). Blockchain Technology and its Potential to benefit public services Provision: A short survey. *Future Internet*, *16*(8), 290. https://doi.org/10.3390/fi16080290

Rejeb, A., Keogh, J. G., Zailani, S., Treiblmaier, H., & Rejeb, K. (2020). Blockchain technology in the food industry: A review of potentials, challenges, and future research directions. Logistics, 4(4), 27. https://doi.org/10.3390/logistics4040027

Shahid, A., Almogren, A., Javaid, N., Al-Zahrani, F. A., Zuair, M., & Alam, M. (2020). Blockchain-based agri-food supply chain: A complete solution. IEEE Access, 8, 69230–69243. https://doi.org/10.1109/ACCESS.2020.2986257

Tian, F. (2016). An agri-food supply chain traceability system for China based on RFID & blockchain technology. *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, 1–6. https://doi.org/10.1109/ICSSSM.2016.7538424

Tripathi, G., Ahad, M. A., & Casalino, G. (2023). A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges. *Decision Analytics Journal*, *9*, 100344. https://doi.org/10.1016/j.dajour.2023.100344