Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning

A PROJECT REPORT

Submitted to a

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITYANANTAPUR, ANANTHAPURAMU

In partial fulfillment of the requirements for the award of thedegree of

BACHELOR OF TECHNOLOGYIN INFORMATION TECHNOLOGY

Submitted by

Y VINAY	**19BF1A1262**
P C V SUBBA REDDY	**19BF1A1243**
CH B KARTHIKEYAN	**19BF1A1213**
S V GURU PRANAV	**19BF1A1256**

Under the Esteemed Guidance of

N.NALINI, M. Tech

Assistant Professor

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

(Permanent Affiliation to JNTUA & Approved by AICTE Recognized under Sections 2(f) and12(B) of UGC act 1956.Accredited by NBA, New Delhi & NAAC Bangalore with 'A' Grade, Tirupati-517507, Chittoor, A.P.)

2019-2023

SRI VENKATESWARA COLLEGE OFENGINEERING

(AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

(Permanent Affiliation to JNTUA & Approved by AICTE Recognized under Sections 2(f) and12(B) of UGC act 1956.Accredited by NBA, New Delhi & NAAC Bangalore with ‘A’Grade, Tirupati-517507, Chittoor, A.P.)

CERTIFICATE

This is to certify that the project report entitled,

“EFFECTIVE MANAGEMENT FOR BLOCKCHAIN-BASED AGRI-FOOD SUPPLY CHAINS USING DEEP REINFORCEMENT LEARNING”

Is a bonafide work done by

Y VINAY	**19BF1A1262**
P C V SUBBAREDDY	**19BF1A1243**
CH B KARTHIKEYAN	**19BF1A1213**
S V GURU PRANAV	**19BF1A1256**

For the partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in INFORMATION TECHNOLOGY, JNTUA, Ananthapuramu.

Project Guide
Mrs. N.NALINI, M.Tech
Assistant Professor
Department of IT
SV College of Engineering.
Tirupati – 517507

Head of the Department
Dr. S. Murali Krishna, B.Tech, M.Tech, Ph.D
Professor
Department of IT
SV College of Engineering.
Tirupati - 517507

Submitted for project viva-voce held on : ________________________________

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project report entitled **" Effective Management for Blockchain based Agri-food supply chain using Deep Reinforcement Learning ",** done by us under the guidance of **Mrs.N.Nalini , Assistant Professor, Department of IT** and is submitted in partial fulfillment of requirements for the award of the degree of **Bachelor of Technology** in **Information Technology .**This report is the result of our own effort and it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

Date :

Place :

Y VINAY	(19BF1A1262)
P C V SUBBA REDDY	(19BF1A1243)
CH B KARTHIKEYAN	(19BF1A1213)
S V GURU PRANAV	(19BF1A1256)

ACKNOWLEDGEMENT

We are thankful to our guide **Mrs.N.Nalini , Assistant Professor, Department Of IT** for her valuable guidance and encouragement. Her helping attitude and suggestions have helped in the successful completion of the project.

We are thankful to project coordinators **Mr. R.Raja Kumar, Senior Assistant Professor,** for his guidance and regular schedules.

We would like to express our gratefulness and sincere thanks to **Dr.S.Murali Krishna, Professor & Head, Department of IT,** for his kind help and encouragement during the course of study and in the successful completion of the project work.

We have a great pleasure in expressing our hearty thanks to our beloved **Principal**

Dr. N. Sudhakar Reddy for spending his valuable time with us to complete the project.

Successful completion of the project cannot be done without proper support and encouragement. We sincerely thank the **Management** for providing all the necessary facilities during course of study.

We would like to express our deep gratitude to all those who helped directly or indirectly to transform an idea in to our working project report.

Y VINAY	(19BF1A1262)
P C V SUBBA REDDY	(19BF1A1243)
CH B KARTHIKEYAN	(19BF1A1213)
S V GURU PRANAV	(19BF1A1256)

CONTENTS

ABSTRACT

In agri-food supply chains (ASCs), consumers pay for agri-food products produced by farmers. During this process, consumers emphasize the importance of agri-food safety while farmers expect to increase their profits. Due to the complexity and dynamics of ASCs, the effective traceability and management for agri-food products face huge challenges. However, most of the existing solutions cannot well meet the requirements of traceability and management in ASCs. To address these challenges, we first design a blockchain-based ASC framework to provide product traceability, which guarantees decentralized security for the agri-food tracing data in ASCs. Next, a Deep Reinforcement learning based Supply Chain Management (DR-SCM) method is proposed to make effective decisions on the production and storage of agri-food products for profit optimization. The extensive simulation experiments are conducted to demonstrate the effectiveness of the proposed blockchain-based framework and the DR-SCM method under different ASC environments. The results show that reliable product traceability is well guaranteed by using the proposed blockchain-based ASC framework. Moreover, the DR-SCM can achieve higher product profits than heuristic and Q-learning methods.

LIST OF ABBREVIATIONS

ASC	:	Agri Supply Chain
DRL	:	Deep Reinforcement Learning
DR-SCM	:	Deep Reinforcement Learning based Supply Chain Management

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.INTRODUCTION

This chapter presents an overview or research work about the Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning . This chapter presents the need and importance of the research problem followed by the scope and significance of the project.

1.1 Background of the study

The problems of agri-food safety and farmer income have receive widespread attentions.The issues of agri-food safety may occur in each part of agri-food supply chains (ASCs), while inefficient management strategies of ASCs would lead to low profits. However, many factors may constrain the normal work of ASCs. First, due to the complex structure of ASCs, it is hard to record the full circulation information of agri-food products while ensuring that the information will never be tampered with. Second, the shift of consumer preferences has become the main barrier of precisely determining the production and storage of agri-food products with the consideration of profit maximization

1.2 Scope of the Study

Although the traditional system would be able to manage supply chain in current market, ithas been noticed that it is difficult to satisfy both the end customers and farmers . the traditional supply chain management techniques have limitations, including a lack of transparency, trust, and traceability. These limitations can lead to inefficiencies, waste, and food safety issues, among other challenges.

1.3 Significance of the Study

The project will enhance the traditional agri food supply chain And effective management system for the agri-food supply chain using blockchain technology and deep reinforcement learning algorithms.It many features that benefit the has significant implications for the food industry, as well as for consumers and society as a whole:

i. **Improved efficiency and effectiveness:** The use of deep reinforcement learning algorithms can optimize the supply chain operations, leading to improved efficiency and effectiveness. This can result in reduced costs, improved delivery times, and enhanced quality of food products.

ii. **Increased transparency and trust:** Blockchain technology provides a transparent and secure way of sharing information among the stakeholders in the supply chain. This can increase trust among the stakeholders, such as farmers, retailers, and consumers, leading to more efficient and reliable transactions**.**

iii. **Improved food safety:** The use of blockchain technology can provide a secure and traceable platform for tracking the origin and quality of the food products, reducing the risks of food safety issues. This can also improve the traceability and recall process in case of any food safety incidents.

CHAPTER-2

LITERATURE SURVEY

2.LITERATURE SURVEY

[1] Wang, Wenbo, et al. "A survey on consensus mechanisms and mining strategy management in blockchain networks." IEEE Access 7 (2019): 22328-22370.

The past decade has witnessed the rapid evolution in blockchain technologies, which has attracted tremendous interests from both the research communities and industries. The blockchain network was originated from the Internet financial sector as a decentralized, immutable ledger system for transactional data ordering. Nowadays, it is envisioned as a powerful backbone/framework for decentralized data processing and data-driven self-organization in flat, open-access networks. In particular, the plausible characteristics of decentralization, immutability, and self-organization are primarily owing to the unique decentralized consensus mechanisms introduced by blockchain networks. This survey is motivated by the lack of a comprehensive literature review on the development of decentralized consensus mechanisms in blockchain networks. In this paper, we provide a systematic vision of the organization of blockchain networks. By emphasizing the unique characteristics of decentralized consensus in blockchain networks, our in-depth review of the state-of-the-art consensus protocols is focused on both the perspective of distributed consensus system design and the perspective of incentive mechanism design. From a game-theoretic point of view, we also provide a thorough review of the strategy adopted for self-organization by the individual nodes in the blockchain backbone networks. Consequently, we provide a comprehensive survey of the emerging applications of blockchain networks in a broad area of telecommunication. We highlight our special interest in how the consensus mechanisms impact these applications. Finally, we discuss several open issues in the protocol design for blockchain consensus and the related potential research directions.

Summary: Wang, Wenbo and team describes in this paper, we provide a systematic vision of the organization of blockchain networks

[2] Ambegaonker, Ajeenkkya, Utkarsh Gautam, and Radha Krishna Rambola. "Efficient approach for Tendering by introducing Blockchain to maintain Security and Reliability." 2018

4th International Conference on Computing Communication and Automation (ICCCA). IEEE, 2018.

The problem with present tendering is its reach which is limited to number of people, though the internet is expanding and tendering is also not far from this, we have some online system for tendering but it is not secure as it should be because tendering has confidential data which is not supposed to be leaked and Blockchain solves that problem efficiently. The motive of this research isto find the better ways for tendering, as tendering is very essential part of businesses and development so improvement of this system leads to better development. Time efficiency, employment, fair system are some of the factors which can be improved by the proposed system of this research.

Summary: Ambegaonker, Ajeenkkya and team working on online system for tendering but it is not secure as it should be because tendering has confidential data which is not supposed to be leaked and Blockchain solves that problem efficiently.

[3] Zheng, Zibin, et al. "An overview of blockchain technology: Architecture, consensus, and future trends." 2017 IEEE international congress on big data (BigData congress). IEEE, 2017.

Blockchain, the foundation of Bitcoin, has received extensive attentions recently. Blockchain serves as an immutable ledger which allows transactions take place in a decentralized manner. Blockchain-based applications are springing up, covering numerous fields including financial services, reputation system and Internet of Things (IoT), and so on. However, there are still many challenges of blockchain technology such as scalability and security problems waiting to be overcome. This paper presents a comprehensive overview on blockchain technology. We provide an overview of blockchain architechture firstly and compare some typical consensus algorithms used in different blockchains. Furthermore, technical challenges and recent advances are briefly listed. We also lay out possible future trends for blockchain.

Summary: Zibin and team provide an overview of blockchain architechture firstly and compare some typical consensus algorithms used in different blockchains.

[4] Cachin, Christian, and Marko Vukolić. "Blockchain consensus protocols in the wild." arXiv preprint arXiv:1707.01873 (2017).

A blockchain is a distributed ledger for recording transactions, maintained by many nodes without central authority through a distributed cryptographic protocol. All nodes validate the information to be appended to the blockchain, and a consensus protocol ensures that the nodes agree on a unique order in which entries are appended. Consensus protocols for tolerating Byzantine faults have received renewed attention because they also address blockchain systems. This work discusses the process of assessing and gaining confidence in the resilience of a consensus protocols exposed to faults and adversarial nodes. We advocate to follow the established practice in cryptography and computer security, relying on public reviews, detailed models, and formal proofs; the designers of several practical systems appear to be unaware of this. Moreover, we review the consensus protocols in some prominent permissioned blockchain platforms with respect to their fault models and resilience against attacks.

Summary: Christian, and team discusses the process of assessing and gaining confidence in the resilience of a consensus protocols exposed to faults and adversarial nodes.

[5] Pilkington, Marc. "Blockchain technology: principles and applications." Research handbook on digital transformations. Edward Elgar Publishing, 2016.

This paper expounds the main principles behind blockchain technology and some of its cutting-edge applications. Firstly, we present the core concepts at the heart of the blockchain, and we discuss the potential risks and drawbacks of public distributed ledgers, and the shift toward hybrid solutions. Secondly, we expose the main features of decentralized public ledger platforms. Thirdly, we show why the blockchain is a disruptive and foundational technology, and fourthly, we sketch out a list of important applications, bearing in mind the most recent evolutions.

Summary: Pilkington and team expose the main features of decentralized public ledger platforms.

CHAPTER-3

PROJECT DESCRIPTION

3.PROJECT DESCRIPTION

This chapter will provide the information about the problem statement of project and the details of Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning.

3.1 Problem Definition

In this task traceability face huge challenges, for example all supply and food manufacturing and quality details stored at centralized single server and some malicious users(internal employees) can collect bribes from cheap quality manufacturer to make their productdetails (fake) as super quality and update centralized server and normal customers will viewsuch fake quality details and trust blindly as there is no one to inform customer about its truequality.

Second challenge is how farmers can maximize their profit by efficiently managing their production as there is no one to inform farmers when to sale and when to start production. To overcome from above two challenges author of this paper introduce concept called DR-SCM(Deep Reinforcement learning based Supply Chain Management) which utilizes two differenttechnologies such as Blockchain and Deep Reinforcement learning.

3.2 Project Details

This Plan Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning is to be made specifically for the purpose of implementing a better agri food supply chain system

As for the management system, the farmer or any corresponding representative can enter the details of the farm that is produced and details like what is the product id and product name and description of the product and give details like what type of user and can save details to the block chain by clicking save asc details button which save the block of information to the block chain with previous and current hash code and form a chain that forms a block chain.

The above mentioned would be possible through these procedures:

1) Save ASC Details in Blockchain: using this module we will take product details from users such as farmers, providers, and processors and then store all this details in Blockchain and will Blockchain will store this data as blocks of transaction and for each block it will generate hash code and verified with previous block hash code.
2) Get ASC Details from Blockchain: using this module all users can read or extract details from Blockchain
3) Deep Reinforcement Learning Simulation: using this module we will feed current agriculture stock data to DRL algorithm and then this algorithm will trained itself on available stock and then predict STOCK manufacturing or SALE to farmer
4) Rewards Graph: for each correct prediction DRL will rewards to its decision and for each wrong prediction DRL will get 0 or penalty deduction and the higher the reward the correct is the prediction. In DRL we will take current stock quantity as STATE and producing or selling STOCK will be action.
5) Blockchain Computation Graph: using this module we will plot Blockchain hash computation time graph
6) Get All Blockchain Products ID: using this module we can get all product ID’s stored in Blockchain.

CHAPTER-4
COMPUTATIONAL ENVIRONMENT

4.COMPUTATIONAL ENVIRONMENT

It is very important to give a general insight into the analysis and requirements of the system. The software and hardware requirements for the development of the system is listed below:

4.1 Software Requirements :

• Operating System	: Linux, Windows 7 or later versions
• Languages	: Python 3.7 or later versions.
• Libraries	: keras, Tkinter, numpy, Matplotlib
• Data base	:Blockchain.

4.2 Hardware Requirements :

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware, the size and the capacity requirements are also important.

• Processor	: Intel I3
• RAM	: 4 GB
• Hard Disk	: 256 GB

4.3 Operating Environment :

The proposed software is to run on decentralized network, The project would also require a high-speed internet connection to ensure the timely and secure transmission of data across the supply chain. Additionally, the project would need to comply with relevant data privacy and security regulations and employ appropriate security measures to protect against data breaches or cyber-attacks.

The project would be operated in a dynamic environment that involves various stakeholders, including farmers, distributors, retailers, and consumers. Therefore, it is essential to have clear communication channels, standard protocols, and efficient workflows to ensure the smooth operation of the project

4.4 External Interface Requirements :

- **User Interfaces :** It has been required that every form's interface should be user friendly and simple to use. Besides, there should be customer or farmer accessing the system through keyboard along with the mouse i.e. keyboard shortcuts.

- **Software Interfaces :** It has been required that there could be a necessity of using the stored data for the prediction and Reinforcement Learning for better prediction of the more profitable to both end users. The more the data we have the better the traning will be done on base of rewarding the best outcome for better results.

- **System Future Requirements :** Other than descriptions provided above, the following features were required by the client:

    - The system should be secured enough to rely on.
    - Users should not be allowed to delete/modify any records.
    - Users should not be allowed to view details of others.

4.5 Non-functional Requirements :

- **Performance Requirements :** As it is going to be used by all the concerned farmers and customers, the system should have a good performance in terms of speed and accuracy. The proposed system should be accurate and fast enoughto handle huge data. It should provide fast computation and visualization of the data.
- **Safety Requirements :** As the software is using hashing algorithms for the storing and processing of the data which is very reliable and no chance of losing data at all as the data is not centralized.

- **Security Requirements :** The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural disaster is known as "System Security". The software should not allow the unauthorized access to any module of the system. Besides, it should maintain the privileges granted to users at various user levels. System Security can be divided into four related issues :

- Security
- Integrity
- Privacy
- Confidentiality

1. **System Security :** It refers to the technical innovations and procedures applied to the hardware and operating systems to protect against deliberate or accidental damage from a defined threat.

2. **Data Security :** It is the protection of data fromloss, disclosure, modification and destruction.

3. **System Integrity :** It refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

4. **Privacy :** It defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unfair or invalid information.

5. **Confidentiality :** It is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

Security in Software :

Security is a critical aspect of effective management of blockchain-based agri-food supply chains using deep reinforcement learning. The use of blockchain technology provides a transparent and tamper-proof platform for data sharing, which enhances the security and trust of the supply chain. However, there are still some security concerns that need to be addressed.

a. Decentralized and Tamper-Proof: The use of blockchain technology provides a decentralized and tamper-proof platform for data sharing, where data is recorded on a distributed ledger and is cryptographically secured using consensus algorithms. This makes it difficult for attackers to manipulate or tamper with the data.

b. Encryption: The project can use encryption to protect sensitive data such as transaction details, personal information, and business secrets.

c. Immutable Audit Trail: The project can maintain an immutable audit trail of all transactions and events in the supply chain. This can help detect and track any suspicious or malicious activity in the system.

d. Network Security: The project can implement network security measures such as firewalls, intrusion detection and prevention systems, and network segmentation to protect the blockchain network from attacks and unauthorized access

4.6 Software Features :

PYTHON:

Python is a high-level, interpreted programming language that is widely used for developing a variety of applications, including web development, data analysis, artificial intelligence, scientific computing, and more. Python was first released in 1991 and has since become one of the most popular programming languages in use today.

Some of the key features of Python include:

1. Easy to Learn: Python has a simple and straightforward syntax that is easy to learn and understand. This makes it an ideal language for beginners and for rapid prototyping.

2. Interpreted Language: Python is an interpreted language, which means that the code is executed line-by-line by an interpreter. This allows for rapid development and testing of code.

3. Dynamic Typing: Python is dynamically typed, which means that variable types are determined at runtime. This allows for more flexibility in programming and makes it easier to write code quickly.

4. Open-Source: Python is an open-source language, which means that the source code is freely available and can be modified and distributed by anyone.

5. Cross-Platform: Python can be run on a variety of platforms, including Windows, Linux, macOS, and more. This makes it easy to develop and deploy Python applications on different platforms.

6. Large Standard Library: Python comes with a large standard library that provides a wide range of modules and functions for common programming tasks, such as file handling, network programming, and more.

7. Object-Oriented: Python is an object-oriented language, which means that it supports object-oriented programming (OOP) concepts such as encapsulation, inheritance, and polymorphism.

8. High-Level Language: Python is a high-level language, which means that it provides abstractions that allow developers to focus on the logic of their code rather than the low-level details of the computer hardware.

BLOCK CHAIN:

Blockchain is a decentralized, digital ledger that is used to record transactions and store data in a secure and transparent manner. It is a revolutionary technology that has gained significant attention in recent years due to its potential to disrupt various industries, including finance, supply chain management, and more.

Some of the main features of blockchain technology include:

1. Decentralization: Blockchain technology is decentralized, meaning that it operates on a network of computers rather than a centralized system. This eliminates the need for intermediaries, such as banks or other financial institutions, and allows for peer-to-peer transactions.

2. Transparency: The ledger in a blockchain is publicly accessible, allowing for transparent tracking of transactions and data. This makes blockchain ideal for industries such as supply chain management, where transparency and traceability are crucial.

3. Security: Blockchain uses cryptography to secure transactions and data, making it difficult for unauthorized parties to access or modify the data stored on the network. This makes it a highly secure platform for data storage and transaction processing.

4. Immutability: Once a transaction or data is recorded on the blockchain, it cannot be modified or deleted. This creates a permanent and tamper-proof record of all transactions and data stored on the network.

5. Efficiency: Blockchain technology can process transactions quickly and efficiently, reducing the time and cost associated with traditional transaction processing methods.

CHAPTER -5

FEASIBILTY STUDY

5.FEASIBILITY STUDY

Feasibility study includes consideration of all the possible ways to provide a solution to the problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements .There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

5.1 Technical Feasibility :

The technical issue usually raised during the feasibility stage of the investigationincludes the following:

- Does the necessary technology exist to do what is suggested?
- Does the proposed equipment's have the technical capacity to hold the data required touse the new system?
- Will proposed system provide adequate response to inquiries, regardless of the numberof users?
- Are there technical guarantees of accuracy,reliability,ease of access and data security?

This includes the study of function, performance and constraints that may affect the ability to achieve an acceptable system. The proposed system will be developed in python completely .The system is developed using Python as the main development language due to client's hardware specification, and blockchian is used to store the data in the ledger which use encrypting techniques to . All the chosen technologies are widely used in the world and are available for free use. These technologies are open source software and does not require any registering or purchasing of any kind.

5.2 Operational Feasibility :

The operational feasibility of the "Effective Management of Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning" project depends on technical expertise in blockchain technology and deep reinforcement learning, integration with existing systems, infrastructure requirements, regulatory and legal compliance, and cost-benefit analysis. The project must be integrated with supply chain management and payment systems while complying with data privacy, intellectual property, and financial regulations. The organization must ensure a robust infrastructure, high-speed internet connectivity, computing resources, and secure storage systems.

5.3 Economic Feasibility :

The economic feasibility depends on several factors. The project can bring economic benefits such as reduced costs through increased efficiency, improved supply chain transparency, and reduced food waste. The implementation cost of the project, including hardware, software, and personnel, should be evaluated against the expected benefits. Additionally, the project's potential revenue streams, such as subscription fees, data analytics, or licensing, should be considered. Overall, the economic feasibility of the project depends on the balance between the costs and the expected benefits, which can be assessed through a thorough cost-benefit analysis.

CHAPTER-6

SYSTEM ANALYSIS

6.SYSTEM ANALYAIS

6.1 Existing System

The existing Traditional agri-food supply chain management typically involves manual processes, limited use of technology, and a lack of integration between different stages of the supply chain. Some of the commonly used systems in traditional agri-food supply chain management include pen and paper, excel sheets, traditional transportation, phone calls and emails which are very in efficient in tracking of supply chain and make problems more complex.

Overall, traditional agri-food supply chain management systems are characterized by a lack of integration and coordination between different stages of the supply chain, which can lead to inefficiencies, delays, and waste. However, it is important to note that many farmers and small-scale producers still rely on these systems due to limited access to technology and resources.

6.1.1 Drawbacks in Existing System

- Lack of Transparency: The agri-food supply chain is often complex and opaque, making it challenging to track the origin, quality, and safety of food products.

- Food Safety Concerns: The agri-food supply chain is prone to food safety issues, such as contamination, spoilage, and adulteration, which can harm public health and reduce consumer trust in the industry.

- Environmental Impact: The agri-food supply chain can have a significant environmental impact due to the use of resources such as water, land, and energy, as well as the production of greenhouse gas emissions.

- Labor Exploitation: The agri-food supply chain can involve labor exploitation, including forced labor, child labor, and poor working conditions, particularly in developing countries.

- Inefficient Distribution: The agri-food supply chain can suffer from inefficiencies, such as overproduction and waste, due to inadequate distribution and storage systems.

6.2 Proposed System

We first design a blockchain-based ASC framework to provide product traceability, which guarantees decentralized security for the agri-food tracing data in ASCs. Next, a Deep Reinforcement learning based Supply Chain Management (DR-SCM) method is proposed to make effective decisions on the production and storage of agri-food products for profit optimization. The extensive simulation experiments are conducted to demonstrate the effectiveness of the proposed blockchain-based framework and the DR-SCM method under different ASC environments. The results show that reliable product traceability is well guaranteed by using the proposed blockchain-based ASC framework. Moreover, the DR-SCM can achieve higher product profits than heuristic and Q-learning methods.

6.2.1 Advantages in Proposed System

- Increased efficiency: The project can improve the efficiency of the agri-food supply chain by optimizing the supply chain operations using deep reinforcement learning algorithms. This can result in reduced costs, improved delivery times, and enhanced quality of food products.

- Enhanced transparency and trust: The project can enhance the transparency and trust among the stakeholders in the agri-food supply chain, such as farmers, retailers, and consumers. This can lead to more efficient and reliable transactions, reducing fraud and increasing the accuracy of information.

- Improved food safety: The project can improve the safety of the food products by providing a secure and traceable platform for tracking the origin and quality of the food products. This can reduce the risks of food safety issues and improve the traceability and recall process in case of any food safety incidents.

- Reduced waste: The project can help to reduce waste in the food industry by optimizing the supply chain operations and ensuring that the right amount of food products are produced and delivered at the right time. This can help to reduce food waste, which is a significant challenge for the food industry.

- Sustainability: The project can contribute to more sustainable practices in the food industry, such as reduced greenhouse gas emissions, more efficient use of resources, and improved environmental practices. This can lead to a more sustainable food industry, which is better for the environment and society as a whole.

CHAPTER-7
SYSTEM DESIG

7.SYSTEM DESIGN

System design is the process of art of defining the architecture, components, modules and data for a system to satisfy specified requirements.

7.1 Data Flow Diagrams :

A data flow diagram is graphical tool used to describe and analyze movement of data through the system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of dataflow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD's is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level.

DFD Symbols :

In the DFD, there are four symbols :

1. A square defines a source or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which information flows.
3. A circle or a bubble represents aprocess that transforms including incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data.

Constructing a DFD :

Several rules of thumb are used in drawing DFD's :

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.

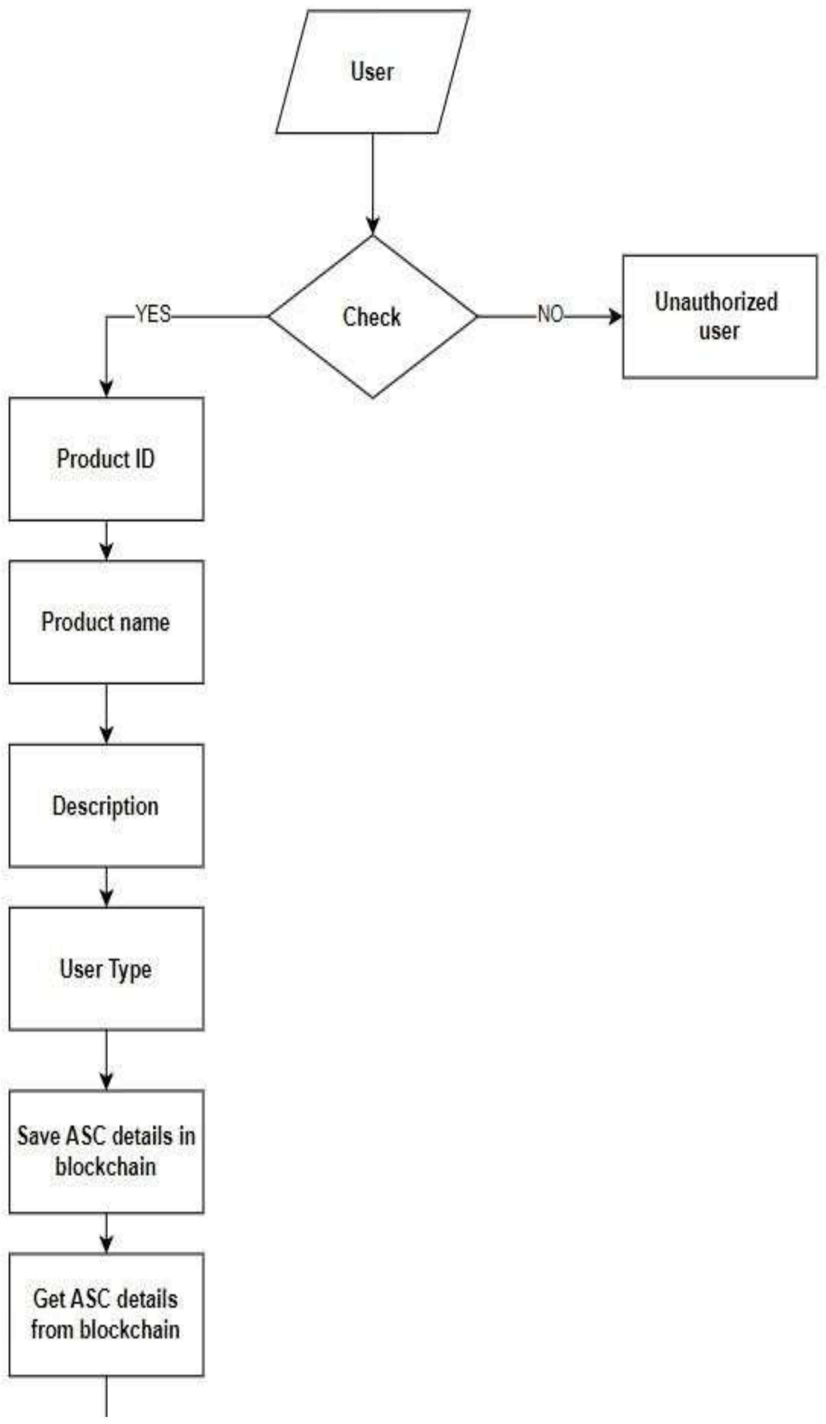
2. The direction of flow is from top to bottom and from left to right. The data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out in the system.

Salient features of DFD's :

1. The DFD shows flow of data, not of control loops and decision are controlled and considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

Data Flow Diagram :



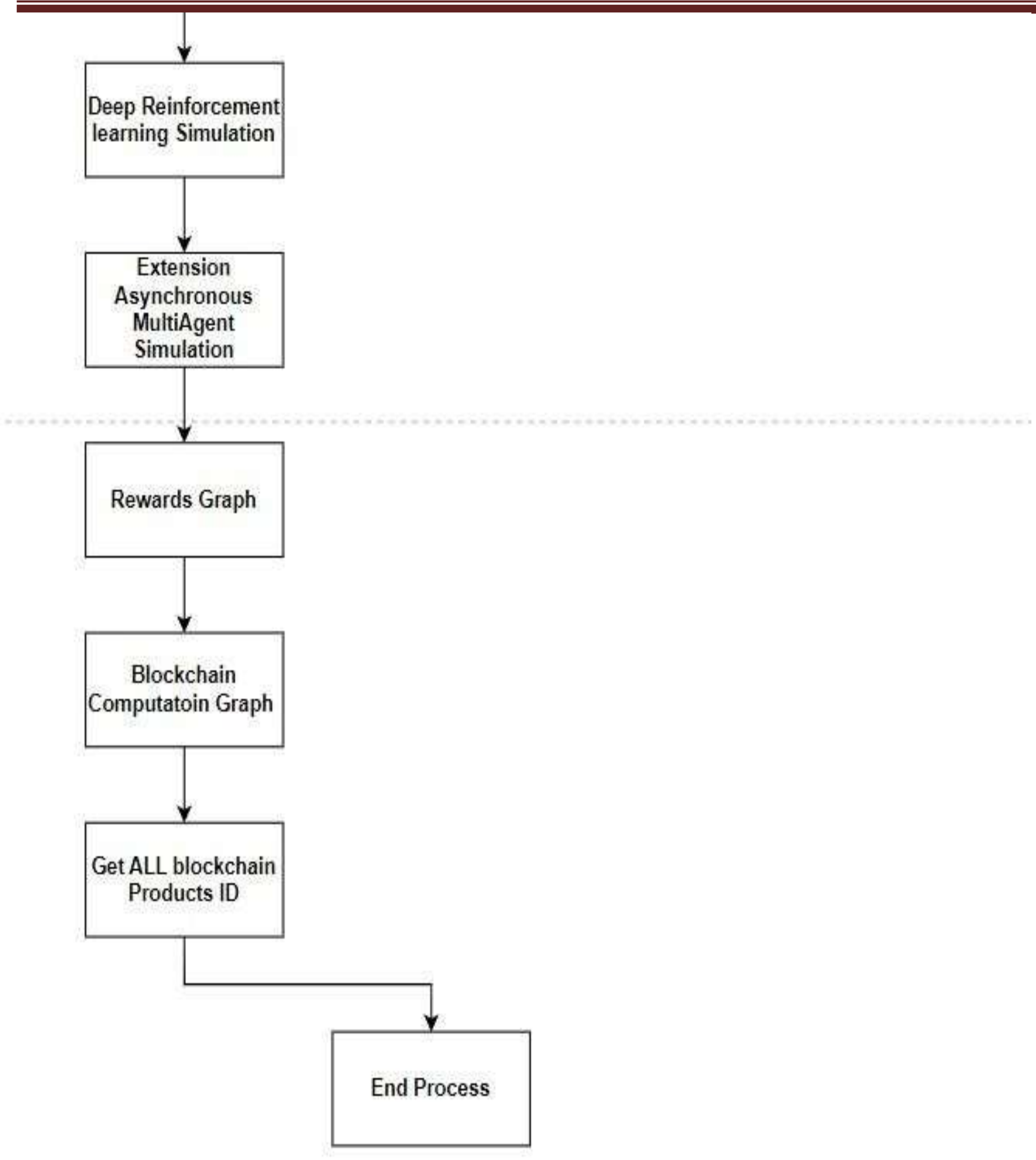

Deep Reinforcement learning Simulation
Extension Asynchronous MultiAgent Simulation
Rewards Graph
Blockchain Computatoin Graph
Get ALL blockchain Products ID
End Process

Fig 7.1.1

7.2 UML Diagrams

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explorepotential designs, and validate the architectural design of the software. System design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

Contents of UML : In general, a UML diagram consists of the following features:

- **Entities:** These may be classes, objects, users or systems behaviours.
- Relationship Lines that model the relationships between entities in the system.
- **Generalization:** A solid line with an arrow that points to a higher abstraction of the present item.
- **Association:** A solid line that represents that one entity uses another entity as part of its behavior.
- **Dependency:** A dotted line with an arrowhead that shows one entity depends on the behavior of another entity.

In this project four basic UML diagrams have been explained:

1) Class Diagram
2) Use Case Diagram
3) Sequence Diagram
4) Activity Diagram

Class Diagram :

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships betweenclasses, not those between specific objects instantiated from those classes. Thus the diagram applies to all the objects in the system.

A class diagram consists of the following features:

- **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifies. Abstract roles of the Class in the system can also be indicated.
- **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Lines that model the relationships between classes and interfaces in the system.
- **Dependency:** A dotted line with an open arrowhead that shows one entity dependson the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter.
- **Aggregation:** Represented by an association line with a hollow diamond at the tail end. An aggregation models the notion that one object uses another object without "owning" it and hus is not responsible for its creation or destruction
- **Inheritance:** A solid line with a solid arrowhead that points from a sub-class to a super class or from a sub-interface to its super-interface.
- **Implementation:** A dotted line with a solid arrowhead that points from a class to the interface that it implements.
- **Composition:** Represented by an association line with a solid diamond at the tail end. A composition models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object. Class Diagram consists of the classes and the objects and the interaction between them. It mainly deals with the interaction between classes in the system, their behavior and properties of the system. Apart from classes this also provides inheritance relationships in the project.

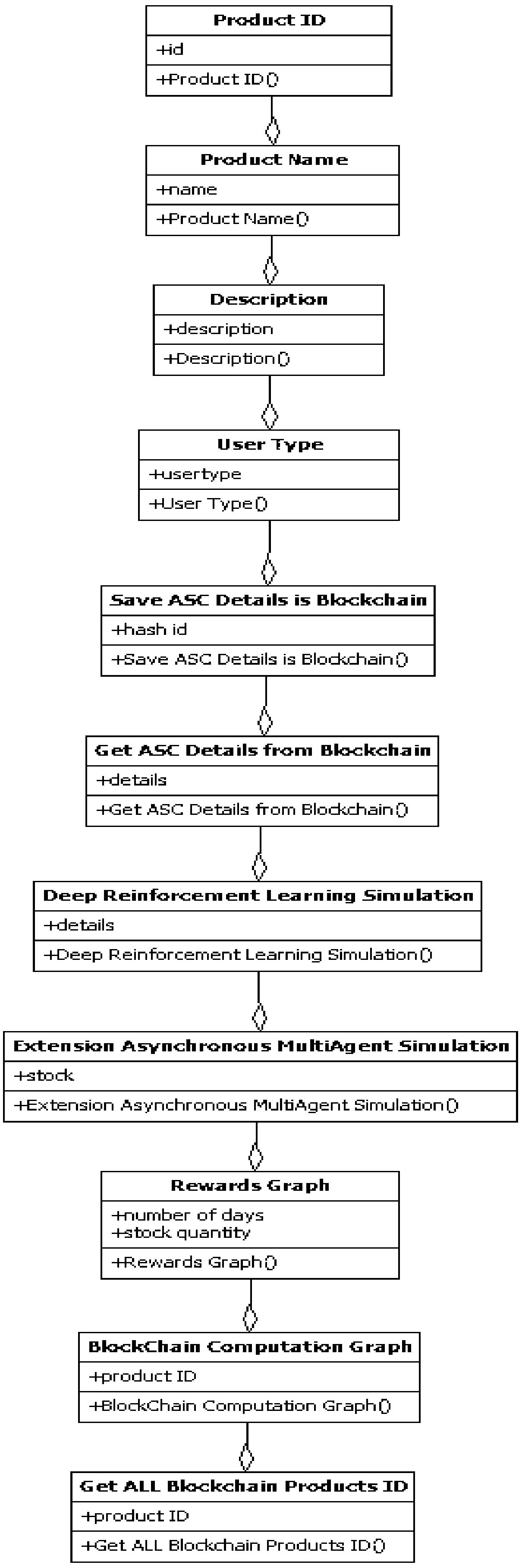

Product ID
+id
+Product ID()
Product Name
+name
+Product Name()
Description
+description
+Description()
User Type
+usertype
+User Type()
Save ASC Details is Blockchain
+hash id
+Save ASC Details is Blockchain()
Get ASC Details from Blockchain
+details
+Get ASC Details from Blockchain()
Deep Reinforcement Learning Simulation
+details
+Deep Reinforcement Learning Simulation()
Extension Asynchronous MultiAgent Simulation
+stock
+Extension Asynchronous MultiAgent Simulation()
Rewards Graph
+number of days
+stock quantity
+Rewards Graph()
BlockChain Computation Graph
+product ID
+BlockChain Computation Graph()
Get ALL Blockchain Products ID
+product ID
+Get ALL Blockchain Products ID()

Fig 7.2.1

Use Case Diagram

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used togetherin a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users.

A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.
The main purpose of a use case diagram is to show what system functions areperformed for which actor. Roles of the actors in the system can be depicted.

Parts of Use cases :

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

- **Actors:** An actor is a person, organization, or external system that plays a role in one or more interactions with the system.
- **System boundary boxes (optional) :** A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not Relationships.
- **Include :** In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case”

Steps To Draw Usecases :

- Identifying Actor
- Identifying Use cases
- Review your use case for completeness

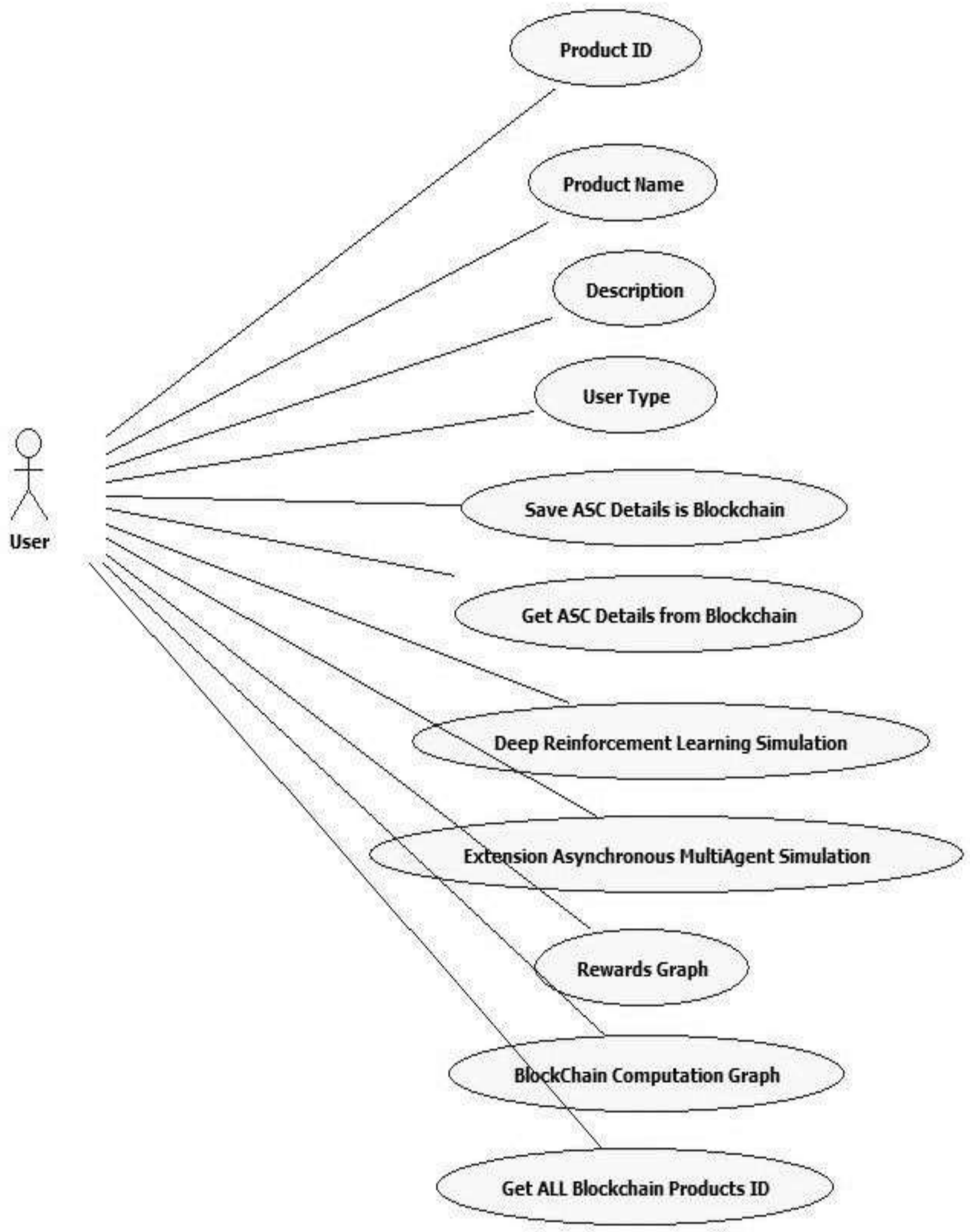

Product ID
Product Name
Description
User Type
User
Save ASC Details is Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Extension Asynchronous MultiAgent Simulation
Rewards Graph
BlockChain Computation Graph
Get ALL Blockchain Products ID

Fig 7.2.2

Activity Diagram :

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by usingdifferent elements like fork, join etc.

The elements of an Activity Diagram are :

- Activities
- Association
- Conditions
- Constraints

The following are the basic notational elements that can be used to make up a diagram:

- **Initial state:** An initial state represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The outgoing transition from the initial vertex may have a behavior, but not a trigger or guard. It is represented by Filled circle, pointing to the initial state.
- **Final state:** A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. It is represented by Hollow circle containing a smaller filled circle, indicating the final state.
- **Rounded rectangle:** It denotes a state. Top of the rectangle contains a name of the state. Can contain a horizontal line in the middle, below which the activities that are done in that state are indicated.
- **Arrow**: It denotes transition. The name of the event (if any) causing this transition labels the arrow body.

Steps to Construct Activity Diagram:

- Identify the preconditions of the workflow.
- Collect the abstractions that are involved in the operations.
- Beginning at the operation’s initial state, specify the activities and actions.
- Use branching to specify conditional paths and iterations.
- Use forking & joining to specify parallel flows of control.

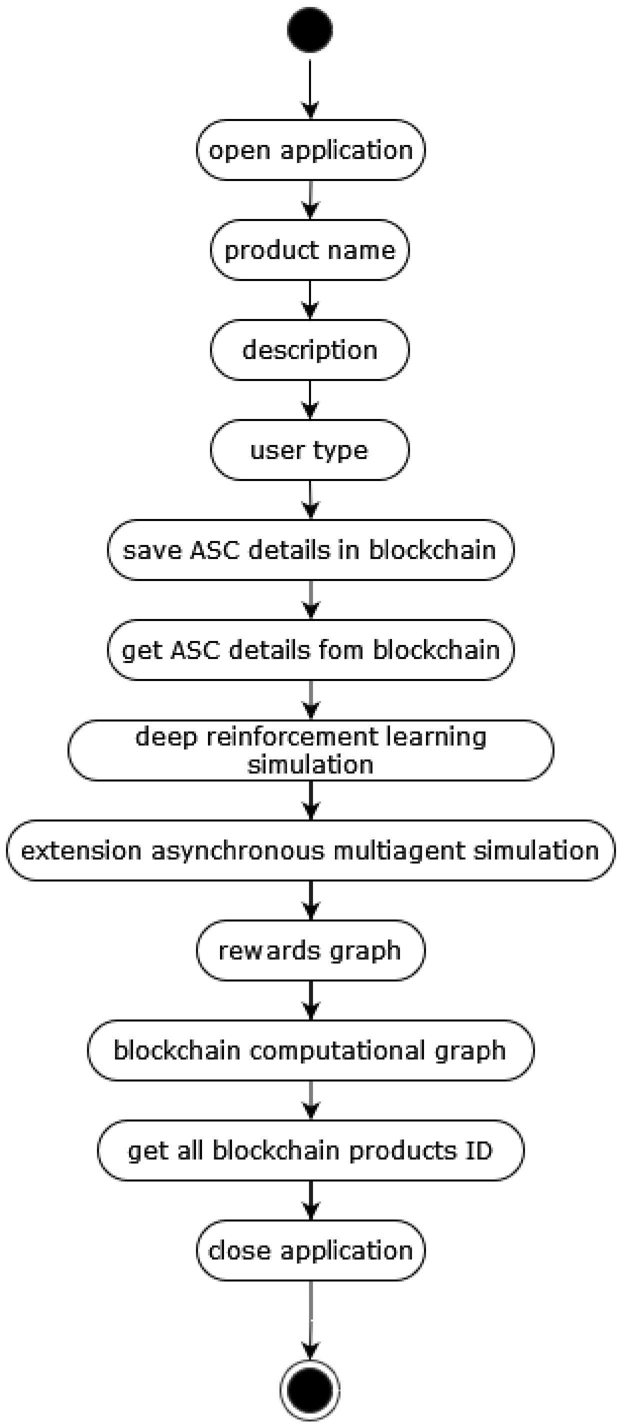

open application
product name
description
user type
save ASC details in blockchain
get ASC details fom blockchain
deep reinforcement learning simulation
extension asynchronous multiagent simulation
rewards graph
blockchain computational graph
get all blockchain products ID
close application

Fig 7.2.3

Sequence Diagram :

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behaviour of a system.

Elements of sequence diagram:

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

- **Lifeline:** A lifeline will generally be generated, and it is a dashed line that sits vertically, and the top will be in the form of a rectangle. This rectangle is used to indicate both the instance and the class. If the lifeline must be used to denote an object, it will be underlined.
- **Messages:** To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it’s solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads withdash arrows are used to represent return messages.
- **Objects:** Objects will also be given the ability to call methods upon themselves, and they can add net activation boxes. Because of this, they can communicate with others to show multiple levels of processing.

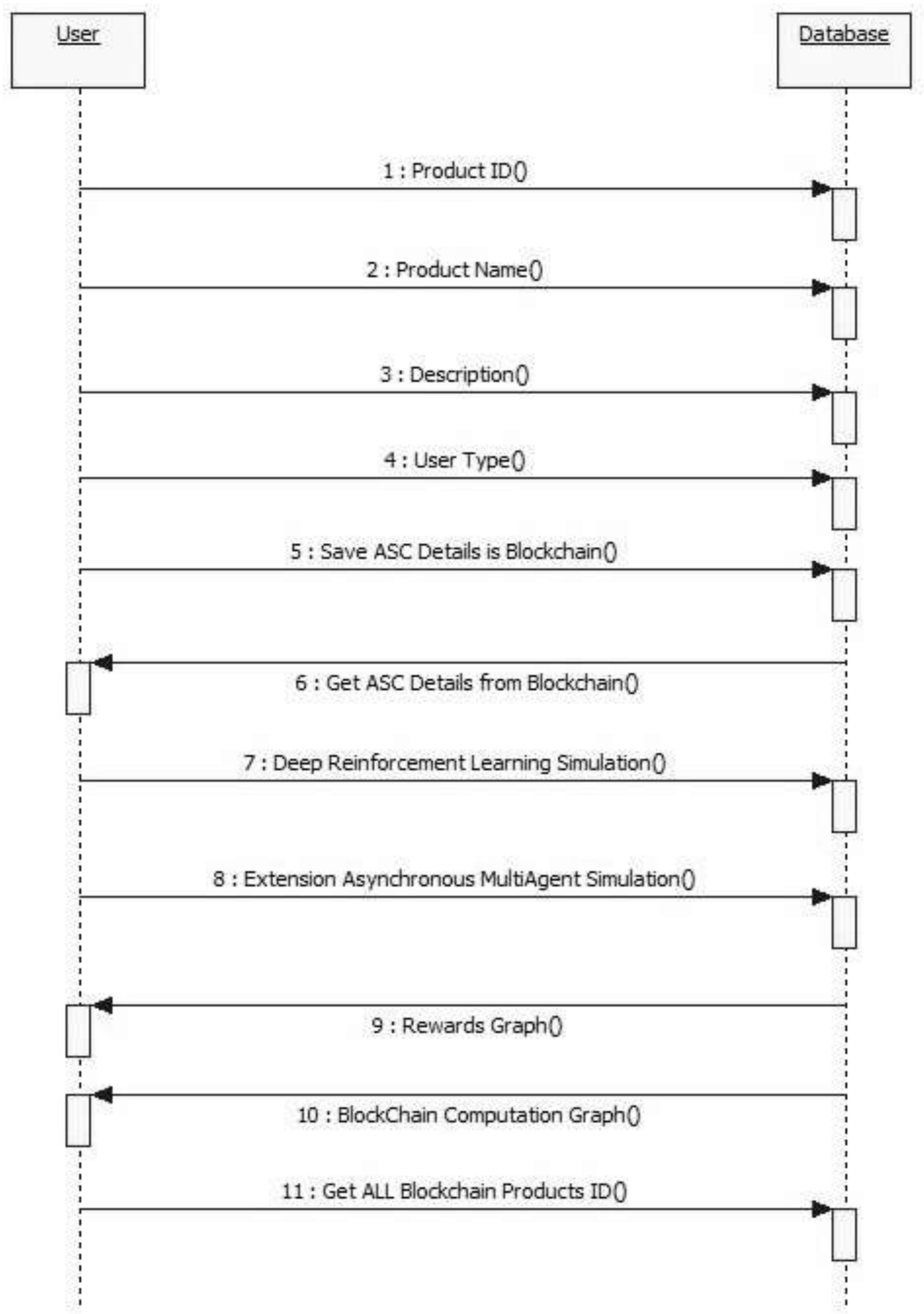

User
Database
1 : Product ID()
2 : Product Name()
3 : Description()
4 : User Type()
5 : Save ASC Details is Blockchain()
6 : Get ASC Details from Blockchain()
7 : Deep Reinforcement Learning Simulation()
8 : Extension Asynchronous MultiAgent Simulation()
9 : Rewards Graph()
10 : BlockChain Computation Graph()
11 : Get ALL Blockchain Products ID()

Fig 7.2.4

CHAPTER-8

SYSTEM IMPLEMENTATION

8.SYSTEM IMPLEMENTATIO

MODULES:

Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning involves several modules that work together to optimize the agri-food supply chain. Some of the key modules involved in the project are :

- Save ASC Details in Blockchain
- Get ASC Details from Blockchain
- Deep Reinforcement Learning Simulation
- Rewards Graph
- Blockchain Computation Graph
- Get All Blockchain Products ID

1) Save ASC Details in Blockchain: using this module we will take product details from users such as farmers, providers, and processors and then store all this details in Blockchain and will Blockchain will store this data as blocks of transaction and for each block it will generate hash code and verified with previous block hash code.
2) Get ASC Details from Blockchain: using this module all users can read or extract details from Blockchain
3) Deep Reinforcement Learning Simulation: using this module we will feed current agriculture stock data to DRL algorithm and then this algorithm will trained itself on available stock and then predict STOCK manufacturing or SALE to farmer
4) Rewards Graph: for each correct prediction DRL will rewards to its decision and for each wrong prediction DRL will get 0 or penalty deduction and the higher the reward the correct is the prediction. In DRL we will take current stock quantity as STATE and producing or selling STOCK will be action.
5) Blockchain Computation Graph: using this module we will plot Blockchain hash computation time graph
6) Get All Blockchain Products ID: using this module we can get all product ID's stored in Blockchain.

CHAPTER-9

TESTING

9.TESTING

A “program unit” stands for a routine or a collection of routines implemented by an individual programmer. It might even be a stand-alone program or a functional unit a larger program.

Unit Testing :

Unit testing is performed prior to integration of the unit into a larger system. It is like coding and debugging ->unit testing ->integration testing. A program unit must be tested for functional tests, performance tests, stress tests and structure tests.

Functional tests refer to executing the code with standard inputs, for which the results will occur within the expected boundaries. Performance test determines the execution time spent in various parts of the unit, response time, device utilization and throughput. Performance testing will help the tuning of the system.

Stress tests drive the system to its limits. They are designed to internationally break the unit. Structure tests verify logical execution along different execution paths. Functional, performance and stress tests are collectively known as “black box testing”. Structure testing is referred to as “white box” or “glass box” testing. Program errors can be classified as missing path errors, computational errors and domain errors.

Even if it looks like all possible execution paths have been tested, there might still exist some more paths. A missing path error occurs, when a branching statement and the associated computations are accidentally omitted. Missing paths can be detected only by functional specifications. A domain error occurs when a program traverses the wrong path because of an incorrect predicate in a branching statement. When a test case fails to detect a computational error there is said to be a coincidental error.

Steps involved during Unit Testing are as follows :

- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Complete code review of the module.
- Actual testing done manually.

- Modifications done for the errors found during testing.
- Prepared the test result scripts.

The unit testing done included the testing of the following items :

- Functionality of the entire module/forms.
- Validations for user input.
- Checking the coding standards to be maintained during coding.
- Testing the module with all the possible test data.
- Testing of the functionality involving all type of calculations.
- Commenting standard in the source files.

After completing the Unit Testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, we integrated the modules one by one and tested the system at each step. This helps in reduction of errors at thetime of system testing.

Integration Testing :

Integration testing strategies include bottom-up (traditional), top-down and sandwich strategies. Bottom-up integration consists of unit testing, followed by testing entire system. Unit testing tries to discover errors in modules. Modules are tested independently in an artificial environment known as “test harness”. Test harnesses provide data environments andcalling sequences for the routines and subsystem that are being tested in isolation.

Disadvantages of bottom-up testing include that harness preparation, which can sometimes take about 50% or more of the coding and debugging effort for a smaller product. After testing all the modules independently and in isolation, they are linked and executed in one single integration run. This is known as “Big bang” approach to integration testing. Isolating sources of errors is difficult in “big bang” approach.

Top-down integration starts with main routine and one or two immediately next lower level routines. After a through checking the top level becomes a test harness to its immediate subordinate routines. Top-down integration offers the following advantages.

System integration is distributed throughout the implementation phase. Modules are integrated as they are developed.

- Top-level interfaces are first test.
- Top-level routines provide a natural test harness for lower-level routines.
- Errors are localized to the new modules and interfaces that are being added.

Though top-down integrations seem to offer better advantages, it may not be applicable in certain situations. Sometimes it may be necessary to test certain low-level modules first. In such situations, a sandwich strategy is preferable. Sandwich integration is mostly top-down, but bottom-up techniques are used on some modules and sub systems. This mixed approach retains the advantages of both strategies.

System Testing :

System testing involves two activities: Integration testing and Acceptance testing. Integration strategy stresses on the order in which modules are written, debugged and unit tested. Acceptance test involves functional tests, performance tests and stress tests to verify requirements fulfillment. System checking checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system.

Steps involved during System Testing are as follows :

- Integration of all the modules in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.

- Recording of all the reproduced errors.
- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.

The System Testing done included the testing of the following items :

- Functionality of the entire system.
- User Interface of the system.
- Testing the dependent modules together with all the possible test data scripts.
- Verification and validation testing.
- Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the acceptance testing. Clients at their end did this and accepted the system with appreciation.

Acceptance Testing :

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to check whether the system implemented satisfies the requirements specifications. Quality assurance people as well as customers may simultaneously develop acceptance tests and run them. In additional to functional and performance tests, stress test are performed to determine the limits/limitations of the system developed. For example, a compiler may be tested for its symbol table overflows or a real- timesystem may be tested for multiple interrupts of different/same priorities.

Acceptance test tools include a test coverage analyzer, and a coding standards checker. Test coverage analyzer records the control paths followed for each test case. A timing analyzer reports the time spent in various regions of the source code under different test cases. Coding standards are stated in the product requirements. Manual inspection is usually not an adequate mechanism from detecting violations of coding standards.

There are some other tests, which fall under special category. They are described below :

- **Peak Load Test :** It determines whether system will handle the volume of activities that occur when the system is at the peak of its processing demand. Foe example, test the system by activating all terminals at the same time.

- **Storage Testing :** It determines the capacity of the system to store transaction data on a disk or in other files.

- **Performance Time Testing :** It determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.

- **Human Factors Testing :** It determines how users will use the system when processing data or preparing reports.

- **Recovery Testing :** This testing determines the ability of user to recover data or restart system after failure. For example, load backup copy of data and resumeprocessing without data or integrity loss.

- **Procedure Testing :** It determines the clarity of documentation on operation and uses of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer.

Testing Objectives :

Testing is vital to the success of the system. System testing is the state of implementation, which ensures that the system works accurately before live operations commence. System testing makes a logical assumption that the system is correct and that the system is correct and that the goals are successfully achieved.

Effective Testing Prerequisites :

Integration testing :

An overall test plan for the project is prepared before the start of coding.

Validation testing :

This project will be tested under this testing sample data and produce the correct sample output.

Recovery testing :

This project will be tested under this testing using correct data input and its product and the correct valid output without any errors.

Test Data and Input :

Taking various types of data we do the above testing. Preparation of test data plays a vital role in system testing. After preparing the test data the system under study is treated using the test data. While testing the system by using the above testing and correctionmethods, the system has been verified and validated by running with both.

- Run with live data.
- Run with test data.

Run with test data :

In the case the system was run with some sample data. Specification testing was also done for each conditions or combinations for conditions.

Run with live data :

The system was tested with the data of the old system for a particular period. Then the new reports were verified by the last one .

CHAPTER-10
SOURCE CODE

10.SAMPLE CODE

Agent :

```
import keras
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.optimizers import Adam

import numpy as np
import random
from collections import deque
class Agent:
    def__init_(self, state_size, is_eval=False, model_name=""):
        self.state_size = state_size # normalized previous days
        self.action_size = 3 # sit, buy, sell
        self.memory = deque(maxlen=1000)
        self.inventory = []
        self.model_name = model_name
        self.is_eval = is_eval

        self.gamma = 0.95
        self.epsilon = 1.0
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995

        self.model = load_model(model_name) if is_eval else self._model()
```

```
def  _model(self):
    model = Sequential()
    model.add(Dense(units=64, input_dim=self.state_size, activation="relu"))
    model.add(Dense(units=32,  activation="relu"))
    model.add(Dense(units=8, activation="relu"))
    model.add(Dense(self.action_size, activation="linear"))
    model.compile(loss="mse", optimizer=Adam(lr=0.001))

    return model
def act(self, state):
    if not self.is_eval and random.random() <= self.epsilon:
        return random.randrange(self.action_size)

    options = self.model.predict(state)
    return np.argmax(options[0])




def expReplay(self, batch_size):
    mini_batch = []
    l = len(self.memory)
    for i in range(l - batch_size + 1, l):
        mini_batch.append(self.memory[i])

    for state, action, reward, next_state, done in mini_batch:
        target = reward
        if not done:
            target = reward + self.gamma * np.amax(self.model.predict(next_state)[0])

        target_f = self.model.predict(state)
        target_f[0][action] = target
        self.model.fit(state, target_f, epochs=1, verbose=0)

    if self.epsilon > self.epsilon_min:
        self.epsilon *= self.epsilon_decay
```

BLOCK:

```
from hashlib import sha256
import json
import time



class Block:
    def__init_(self, index, transactions, timestamp, previous_hash):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = 0




    def compute_hash(self):
        """
        A function that return the hash of the block contents.
        """
        block_string = json.dumps(self._dict_, sort_keys=True)
        return sha256(block_string.encode()).hexdigest()
```

BLOCK CHAIN :

```
from hashlib import sha256
import json
import time
import pickle
from datetime import datetime
import random
import base64
from Block import *

class Blockchain:
    # difficulty of our PoW algorithm
    difficulty = 2 #using difficulty 2 computation

    def___init_(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()
        self.peer = []
        self.translist = []

    def create_genesis_block(self): #create genesis block
        genesis_block = Block(0, [], time.time(), "0")
        genesis_block.hash = genesis_block.compute_hash()
        self.chain.append(genesis_block)

    @property
    def last_block(self):
        return self.chain[-1]

    def add_block(self, block, proof): #adding data to block by computing new and previous
hashes
        previous_hash = self.last_block.hash

        if previous_hash != block.previous_hash:
            return False

        if not self.is_valid_proof(block, proof):
            return False

        block.hash = proof
        #print("main "+str(block.hash))
        self.chain.append(block)
        return True




    def is_valid_proof(self, block, block_hash): #proof of work
        return (block_hash.startswith('0' * Blockchain.difficulty) and block_hash ==
block.compute_hash())

    def proof_of_work(self, block): #proof of work
        block.nonce = 0

        computed_hash = block.compute_hash()
        while not computed_hash.startswith('0' * Blockchain.difficulty):
            block.nonce += 1
            computed_hash = block.compute_hash()

        return computed_hash


    def add_new_transaction(self, transaction):
        self.unconfirmed_transactions.append(transaction)


    def addPeer(self, peer_details):
        self.peer.append(peer_details)
```

```
def addTransaction(self,trans_details): #add transaction
    self.translist.append(trans_details)

def mine(self):#mine transaction
    if not self.unconfirmed_transactions:
        return False



    last_block = self.last_block



    new_block = Block(index=last_block.index + 1,
                      transactions=self.unconfirmed_transactions,
                      timestamp=time.time(),
                      previous_hash=last_block.hash)

    proof = self.proof_of_work(new_block)
    self.add_block(new_block, proof)

    self.unconfirmed_transactions = []
    return new_block.index



def save_object(self,obj, filename):
    with open(filename, 'wb') as output:
        pickle.dump(obj, output, pickle.HIGHEST_PROTOCOL)
```

MAIN :

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
from Block import *
from Blockchain import *
from hashlib import sha256
import os
from tkinter import ttk
import time


from keras.models import load_model
from Agent import *
import math
import numpy as np
import random
from collections import deque
import sys
import matplotlib.pyplot as plt



main = Tk()
main.title("Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep
Reinforcement Learning")
main.geometry("1300x1200")


global filename
```

```
compute_time = []
purchase_arr = []
sold_arr = []
rewards = []
qrewards = []
extension_rewards = []

blockchain = Blockchain()
if os.path.exists('blockchain_contract.txt'):
    with open('blockchain_contract.txt', 'rb') as fileinput:
        blockchain = pickle.load(fileinput)
    fileinput.close()



def saveASCDetails():
    global  filename
    global compute_time
    text.delete('1.0', END)
    pid = tf1.get()
    pname = tf2.get()
    desc = tf3.get()
    usertype = userlist.get()
    if len(pid) > 0 and len(pname) > 0 and len(desc) > 0:
        start = time.time()
        data = pid+"#"+pname+"#"+desc+"#"+usertype
        blockchain.add_new_transaction(data)
        hash = blockchain.mine()
        b = blockchain.chain[len(blockchain.chain)-1]
        text.insert(END,"Blockchain Previous Hash : "+str(b.previous_hash)+"\nBlock No :
"+str(b.index)+"\nCurrent Hash : "+str(b.hash)+"\n")
        text.insert(END,"Details saved in blockchain\n\n")
        blockchain.save_object(blockchain,'blockchain_contract.txt')
        tf1.delete(0, END)
        tf2.delete(0, END)
        tf3.delete(0, END)
        end = time.time()
        compute_time.append((end - start))
    else:
        text.insert(END,"Please fill all details")



def           getProduct():
    text.delete('1.0', END)
    for i in range(len(blockchain.chain)):
        if i > 0:
            b = blockchain.chain[i]
            data = b.transactions[0]
            arr = data.split("#")
            text.insert(END,"Available    Product    ID    :    "+arr[0]+"\n")
            text.insert(END,"Product Storage Hash ID : "+str(b.hash)+"\n\n")



def getASCDetails():
    text.delete('1.0', END)
    pid = tf1.get()
    flag = False
    for i in range(len(blockchain.chain)):
        if i > 0:
            b = blockchain.chain[i]
            data = b.transactions[0]
            arr = data.split("#")
            if arr[0] == pid:
                text.insert(END,"Product ID    : "+arr[0]+"\n")
                text.insert(END,"Product Name : "+arr[1]+"\n")
                text.insert(END,"Description : "+arr[2]+"\n")
                text.insert(END,"User Type     : "+arr[3]+"\n\n")
                flag = True
    if flag == False:
        text.insert(END,"No records found for given ID")




def formatQuantity(qty):
    return str(qty)


def getStockQuantity():
    vector = []
    index = 0
    with open('Dataset/agriculture_stock.csv', "r") as fileData:
        for line in fileData:
            if index > 0:
                line = line.strip('\n')
                line = line.strip()
                arr = line.split(",")
                value = arr[5]
                value = value[1:len(value)-1]
                vector.append(float(value))
            index = 1
    return vector


def calculateSigmoid(gamma_value):
    if gamma_value < 0:
        return 1 - 1/(1 + math.exp(gamma_value))
    else:
        return 1/(1 + math.exp(-gamma_value))
```

```
def getAgentState(values, index, total):
    data = index - total + 1
    block = values[data:index + 1] if data >= 0 else -data * [values[0]] + values[0:index + 1]
    result = []
    for i in range(total - 2):
        result.append(calculateSigmoid(block[i + 1] - block[i]))
    return np.array([result])

def DRLSimulation():
    global purchase_arr
    global sold_arr
    global rewards
    global qrewards

    text.delete('1.0', END)
    windowSize = 10
    episodeCount = 1000
    windowSize = int(windowSize)
    episodeCount = int(episodeCount)
    count = 0
    agent = Agent(windowSize)
    quantity = getStockQuantity()
    length = len(quantity) - 1
    batchSize = 32
    modelName = 'models/model_ep500'
    model = load_model(modelName)
    windowSize = model.layers[0].input.shape.as_list()[1]
    state = getAgentState(quantity, 0, windowSize + 1)
    cuttent_Quantity = 0
    agent.inventory = []
    purchase_arr.clear()
    sold_arr.clear()
    rewards.clear()
    qrewards.clear()
    qreward = 0


    for t in range(length):
        action = agent.act(state)
        nextState = getAgentState(quantity, t + 1, windowSize + 1)
        rewardValue = 0

        if action == 1: # buy
            agent.inventory.append(quantity[t])
            text.insert(END,"Start Producing Stock : " + formatQuantity(quantity[t])+"\n")
            purchase_arr.append(quantity[t])

        elif action == 2 and len(agent.inventory) > 0:
            old_quantity = agent.inventory.pop(0)
            rewardValue = max(quantity[t] - old_quantity, 0)
            qreward = max(quantity[t] - old_quantity, 0) - 15
            cuttent_Quantity = old_quantity - quantity[t]

            if  old_quantity > quantity[t]:
                text.insert(END,"You can Sale sufficient stock available : " + formatQuantity(quantity[t])
+ "| Current Available : " + formatQuantity(old_quantity)+" | After Sale Available : " +
formatQuantity(old_quantity - quantity[t])+"\n")
                sold_arr.append(old_quantity - quantity[t])

            else:
                text.insert(END,"You can Sale sufficient stock available : " + formatQuantity(quantity[t])
+ "| Current Available : " + formatQuantity(old_quantity)+" | After Sale Available : " +
formatQuantity(quantity[t] - old_quantity)+"\n")
                sold_arr.append(quantity[t] - old_quantity)
        done = True if t == length - 1 else False
        if rewardValue != 0:
            rewards.append(rewardValue)
            qrewards.append(qreward)
        agent.memory.append((state, action, rewardValue, nextState, done))
        state = nextState

        if done:
            text.insert(END,"\n..........................................\n")
            text.insert(END,"Current Available Quantity : " + formatQuantity(cuttent_Quantity)+"\n")
            print("..........................................")

    purchase_arr = np.asarray(purchase_arr)
    sold_arr = np.asarray(sold_arr)
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Days')
    plt.ylabel('Purchase/Sales Count')
    plt.plot(purchase_arr, 'ro-', color = 'green')
    plt.plot(sold_arr, 'ro-', color = 'blue')
    plt.legend(['Available Stock', 'Sold Stock'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('Stock of Factory & Retailers by using DR-SCM')
    plt.show()



def rewardGraph():
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Episodes')
    plt.ylabel('Reward Values')
    plt.plot(rewards, 'ro-', color = 'green')
    plt.plot(qrewards, 'ro-', color = 'blue')
    plt.plot(extension_rewards, 'ro-', color = 'yellow')
    plt.legend(['DR-SCM Rewards', 'QLearning Rewards','Extension MultiAgent Rewards'],
loc='upper left')



    #plt.xticks(wordloss.index)
    plt.title('Stock of Factory & Retailers by using DR-SCM')
    plt.show()


def graph():
    X = []
    for i in range(len(compute_time)):
        X.append("Block "+str((i+1)))
    height = compute_time
    bars = X
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.title("Blockchain Hashes Computation Graph")
    plt.show()


def runMultiAgent(current_reward):
    global extension_rewards
    windowSize = 10
    episodeCount = 1000
    windowSize = int(windowSize)
    episodeCount = int(episodeCount)
    count = 0
    agent = Agent(windowSize)
    quantity = getStockQuantity()
```

```
length = len(quantity) - 1
batchSize = 32
modelName = 'models/model_ep500'
model = load_model(modelName)
windowSize = model.layers[0].input.shape.as_list()[1]
state = getAgentState(quantity, 0, windowSize + 1)
cuttent_Quantity = 0
agent.inventory = []
multi_agent = True
purchase_arr = []
sold_arr = []

while multi_agent == True:
    for t in range(length):
        action = agent.act(state)
        nextState = getAgentState(quantity, t + 1, windowSize + 1)
        rewardValue = 0

        if action == 1: # buy
            agent.inventory.append(quantity[t])
            purchase_arr.append(quantity[t])




        elif action == 2 and len(agent.inventory) > 0:
            old_quantity = agent.inventory.pop(0)
            rewardValue = max(quantity[t] - old_quantity, 0)
            cuttent_Quantity = old_quantity - quantity[t]
            if old_quantity > quantity[t]:
                sold_arr.append(old_quantity - quantity[t])
            else:
                sold_arr.append(quantity[t] - old_quantity)
        done = True if t == length - 1 else False
        print(str(rewardValue)+" "+str(current_reward))
        if rewardValue > current_reward and multi_agent == True:
            extension_rewards.append(rewardValue)
            multi_agent = False
        agent.memory.append((state, action, rewardValue, nextState, done))
        state = nextState


def extensionSimulation():
    global rewards, extension_rewards
    start = 0
    text.delete('1.0', END)
    while start < len(rewards):
        runMultiAgent(rewards[start])
        start = start + 1
    text.insert(END,"Reward earned by Extension Multiagent: "+str(extension_rewards))


font = ('times', 15, 'bold')
title = Label(main, text='Effective Management for Blockchain-Based Agri-Food Supply Chains
Using Deep Reinforcement Learning')

title.config(bg='bisque', fg='purple1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)


font1 = ('times', 13, 'bold')


l1 = Label(main, text='Product ID :')
l1.config(font=font1)
l1.place(x=50,y=100)
```

```
tf1 = Entry(main,width=20)
tf1.config(font=font1)
tf1.place(x=180,y=100)

l2 = Label(main, text='Product Name :')
l2.config(font=font1)
l2.place(x=50,y=150)

tf2 = Entry(main,width=20)
tf2.config(font=font1)
tf2.place(x=180,y=150)

l3 = Label(main, text='Description :')
l3.config(font=font1)
l3.place(x=50,y=200)

tf3 = Entry(main,width=80)
tf3.config(font=font1)
tf3.place(x=180,y=200)

l3 = Label(main, text='User Type :')
l3.config(font=font1)
l3.place(x=50,y=250)



user = ['Providers','Farmers','Processors','Distributors','Retailers','Consumers']
userlist = ttk.Combobox(main,values=user,postcommand=lambda: userlist.configure(values=user))
userlist.place(x=180,y=250)
userlist.current(0)
userlist.config(font=font1)

saveButton = Button(main, text="Save ASC Details in Blockchain", command=saveASCDetails)
saveButton.place(x=50,y=300)
saveButton.config(font=font1)
```

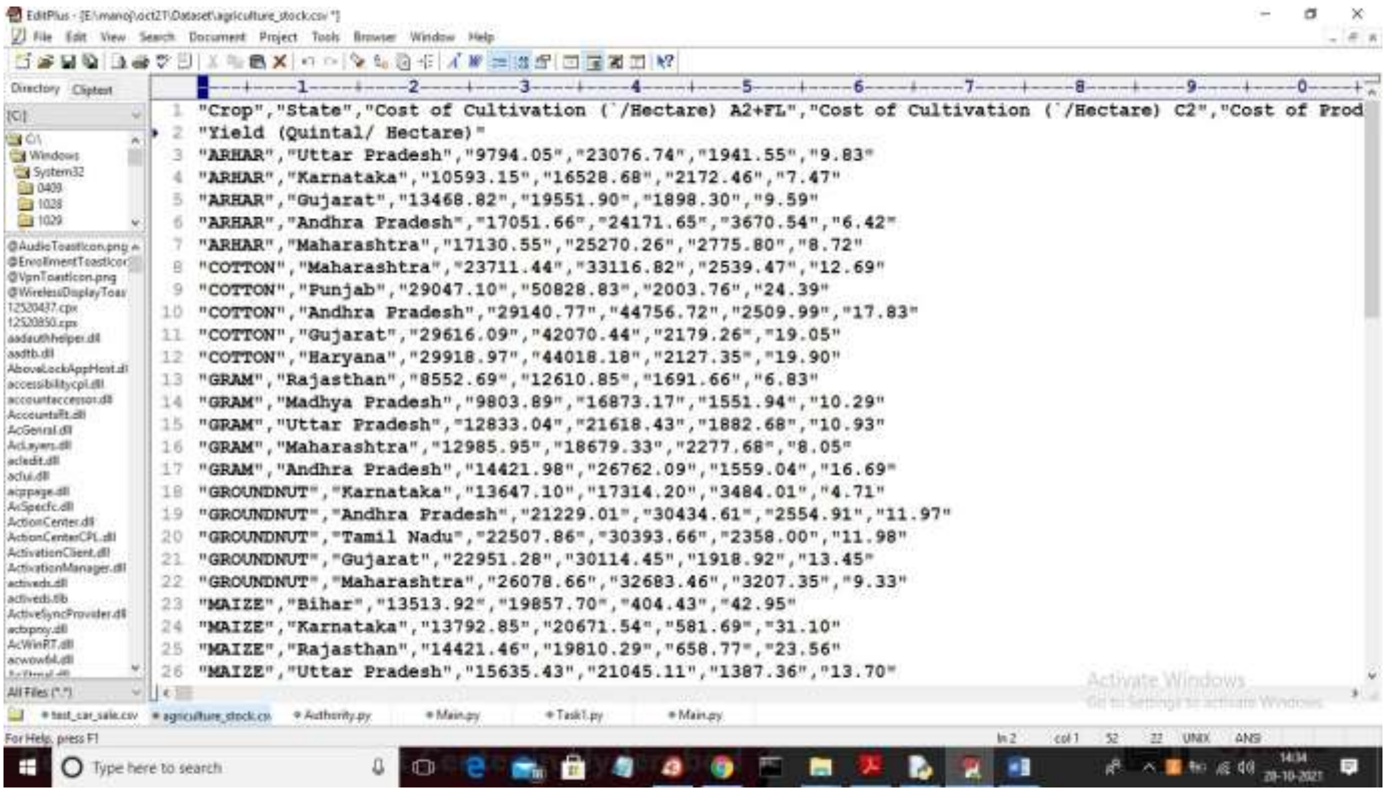
```
verifyButton = Button(main, text="Get ASC Details from Blockchain", command=getASCDetails)
verifyButton.place(x=330,y=300)
verifyButton.config(font=font1)


drlButton = Button(main, text="Deep Reinforcement Learning Simulation",
command=DRLSimulation)
drlButton.place(x=625,y=300)
drlButton.config(font=font1)


extButton = Button(main, text="Extension Asynchronous MultiAgent Simulation",
command=extensionSimulation)
extButton.place(x=985,y=300)
extButton.config(font=font1)



rewardsButton = Button(main, text="Rewards Graph", command=rewardGraph)
rewardsButton.place(x=50,y=350)
rewardsButton.config(font=font1)


graphButton = Button(main, text="Blockchain Computation Graph", command=graph)
graphButton.place(x=330,y=350)
graphButton.config(font=font1)


getButton = Button(main, text="Get All Blockchain Products ID", command=getProduct)
getButton.place(x=625, y=350)


getButton.config(font=font1)
```

```
font1 = ('times', 13, 'bold')
text=Text(main,height=15,width=120)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=400)
text.config(font=font1)


main.config(bg='cornflower blue')
main.mainloop()
```

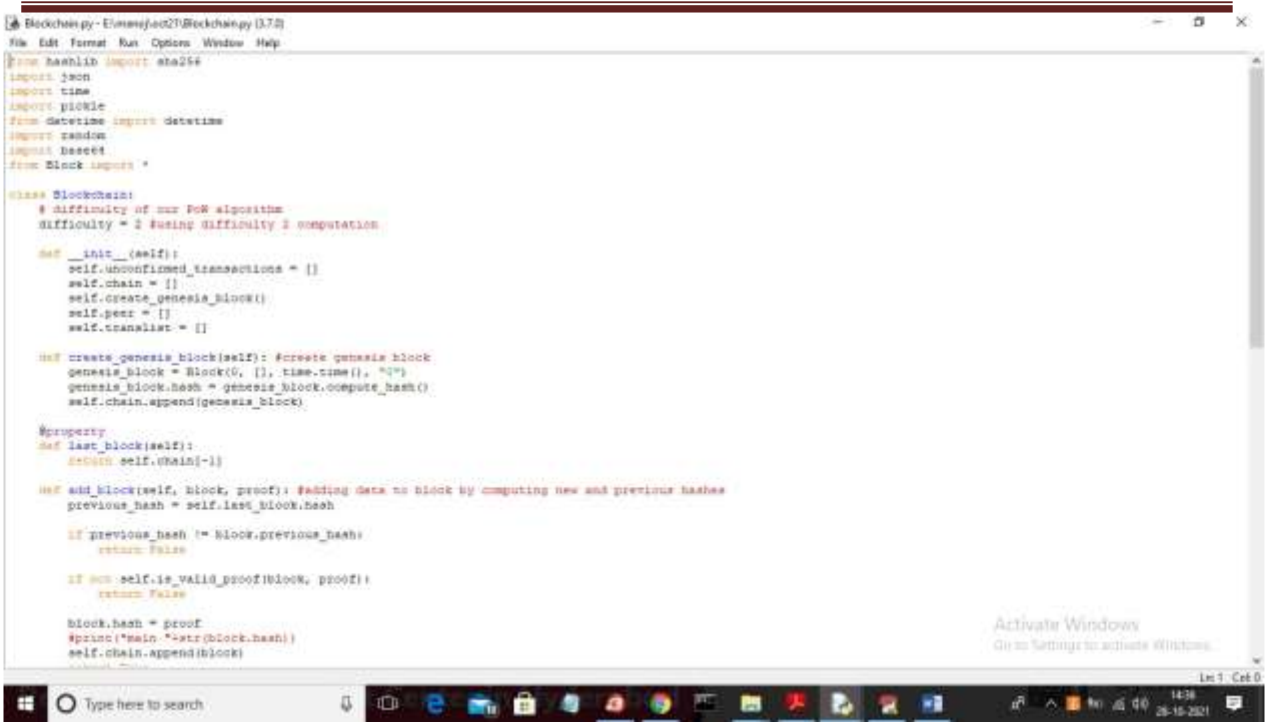
CHAPTER-11
SCREEN LAYOUTS

11.SCREEN LAYOUTS

TRANING DRL ALGORITHM USING STOCK DATA:

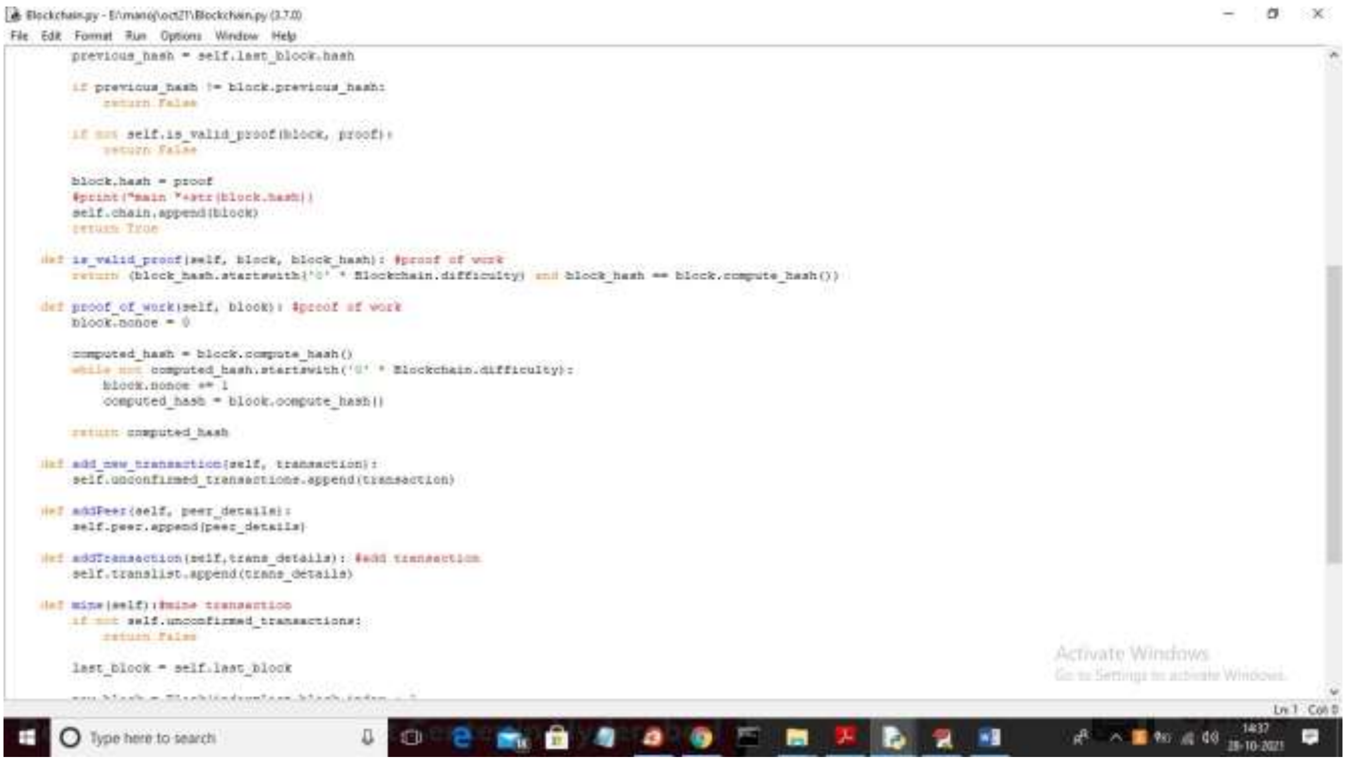

EditPlus - [E:\manoj\oct21\Dataset\agriculture_stock.csv *]
File Edit View Search Document Project Tools Browser Window Help
Directory Cliptext
agriculture_stock.csv
Authority.py
Main.py
Task1.py
Main.py
Activate Windows

```
"Crop","State","Cost of Cultivation (`/Hectare) A2+FL","Cost of Cultivation (`/Hectare) C2","Cost of Prod
"Yield (Quintal/ Hectare)"
"ARHAR","Uttar Pradesh","9794.05","23076.74","1941.55","9.83"
"ARHAR","Karnataka","10593.15","16528.68","2172.46","7.47"
"ARHAR","Gujarat","13468.82","19551.90","1898.30","9.59"
"ARHAR","Andhra Pradesh","17051.66","24171.65","3670.54","6.42"
"ARHAR","Maharashtra","17130.55","25270.26","2775.80","8.72"
"COTTON","Maharashtra","23711.44","33116.82","2539.47","12.69"
"COTTON","Punjab","29047.10","50828.83","2003.76","24.39"
"COTTON","Andhra Pradesh","29140.77","44756.72","2509.99","17.83"
"COTTON","Gujarat","29616.09","42070.44","2179.26","19.05"
"COTTON","Haryana","29918.97","44018.18","2127.35","19.90"
"GRAM","Rajasthan","8552.69","12610.85","1691.66","6.83"
"GRAM","Madhya Pradesh","9803.89","16873.17","1551.94","10.29"
"GRAM","Uttar Pradesh","12833.04","21618.43","1882.68","10.93"
"GRAM","Maharashtra","12985.95","18679.33","2277.68","8.05"
"GRAM","Andhra Pradesh","14421.98","26762.09","1559.04","16.69"
"GROUNDNUT","Karnataka","13647.10","17314.20","3484.01","4.71"
"GROUNDNUT","Andhra Pradesh","21229.01","30434.61","2554.91","11.97"
"GROUNDNUT","Tamil Nadu","22507.86","30393.66","2358.00","11.98"
"GROUNDNUT","Gujarat","22951.28","30114.45","1918.92","13.45"
"GROUNDNUT","Maharashtra","26078.66","32683.46","3207.35","9.33"
"MAIZE","Bihar","13513.92","19857.70","404.43","42.95"
"MAIZE","Karnataka","13792.85","20671.54","581.69","31.10"
"MAIZE","Rajasthan","14421.46","19810.29","658.77","23.56"
"MAIZE","Uttar Pradesh","15635.43","21045.11","1387.36","13.70"
```



```
from hashlib import sha256
import json
import time
import pickle
from datetime import datetime
import random
import base64
from Block import *

class Blockchain:
    # difficulty of our PoW algorithm
    difficulty = 2 #using difficulty 2 computation

    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()
        self.peer = []
        self.translist = []

    def create_genesis_block(self): #create genesis block
        genesis_block = Block(0, [], time.time(), "0")
        genesis_block.hash = genesis_block.compute_hash()
        self.chain.append(genesis_block)

    @property
    def last_block(self):
        return self.chain[-1]

    def add_block(self, block, proof): #adding data to block by computing new and previous hashes
        previous_hash = self.last_block.hash

        if previous_hash != block.previous_hash:
            return False

        if not self.is_valid_proof(block, proof):
            return False

        block.hash = proof
        #print("main "+str(block.hash))
        self.chain.append(block)
```

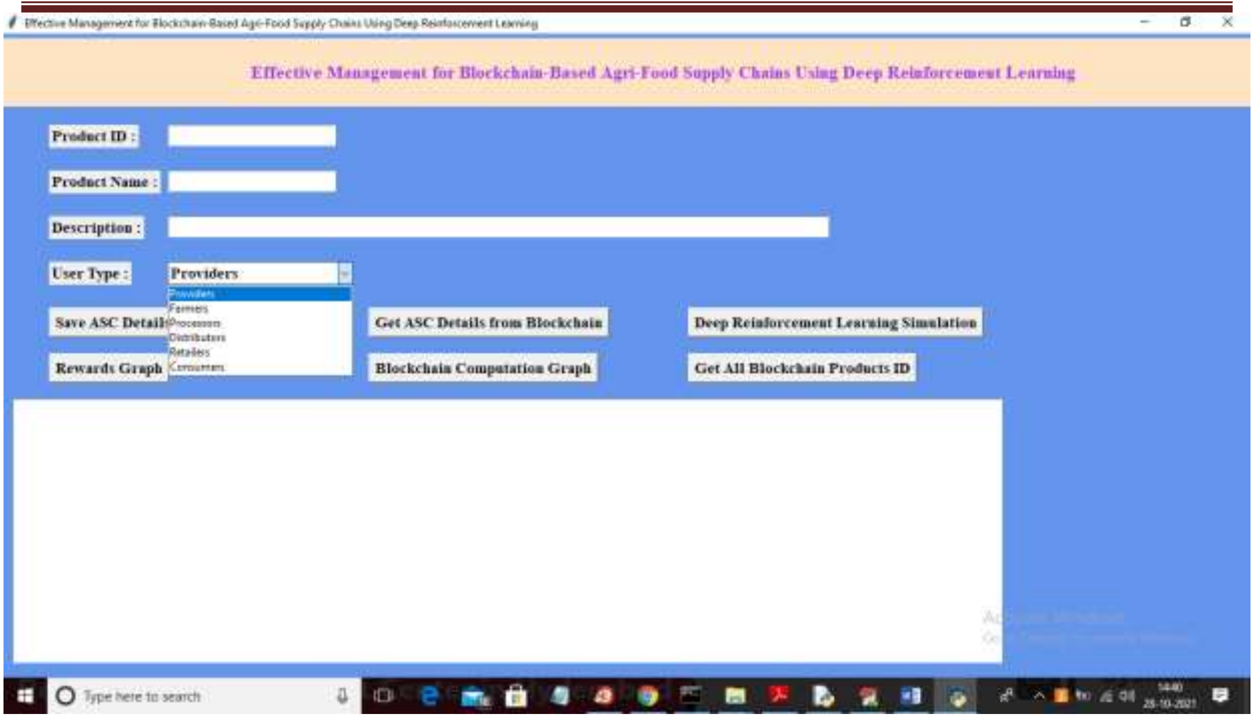
In above screen read red colour comments to know about Blockchain transaction storage



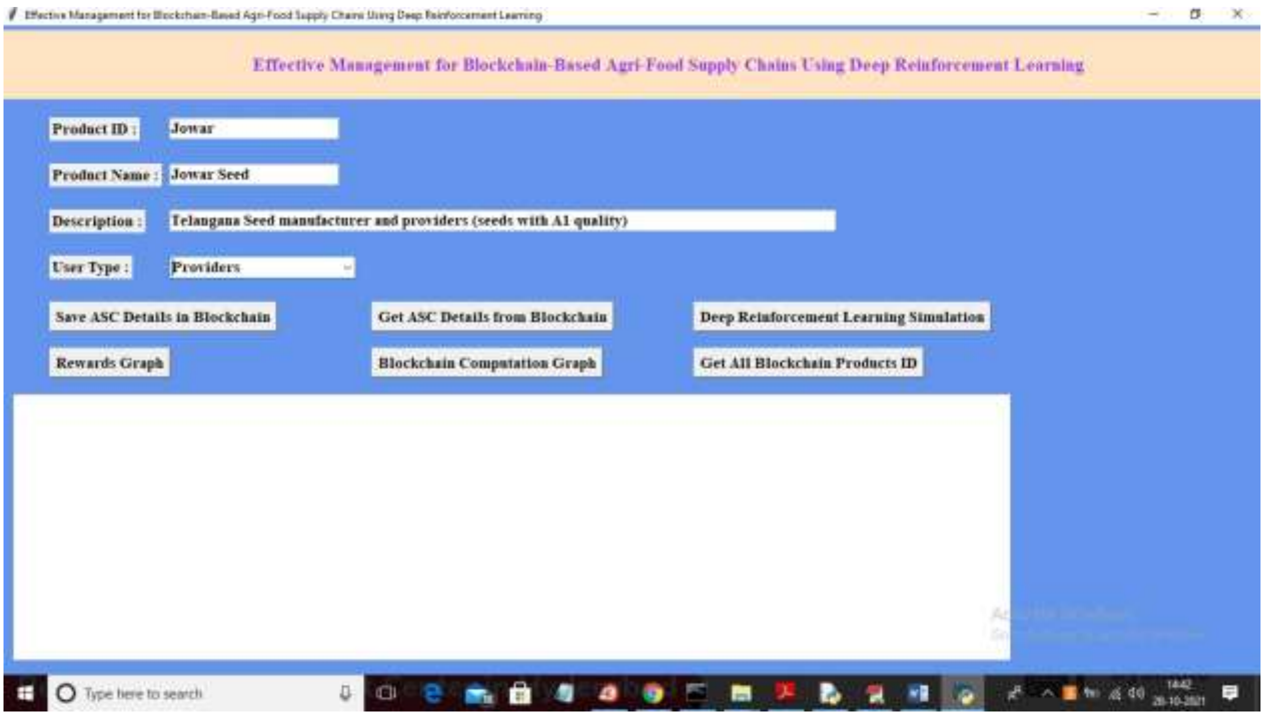
```
        previous_hash = self.last_block.hash

        if previous_hash != block.previous_hash:
            return False

        if not self.is_valid_proof(block, proof):
            return False

        block.hash = proof
        #print("main "+str(block.hash))
        self.chain.append(block)
        return True

    def is_valid_proof(self, block, block_hash): #proof of work
        return (block_hash.startswith('0' * Blockchain.difficulty) and block_hash == block.compute_hash())

    def proof_of_work(self, block): #proof of work
        block.nonce = 0

        computed_hash = block.compute_hash()
        while not computed_hash.startswith('0' * Blockchain.difficulty):
            block.nonce += 1
            computed_hash = block.compute_hash()

        return computed_hash

    def add_new_transaction(self, transaction):
        self.unconfirmed_transactions.append(transaction)

    def addPeer(self, peer_details):
        self.peer.append(peer_details)

    def addTransaction(self,trans_details): #add transaction
        self.translist.append(trans_details)

    def mine(self):#mine transaction
        if not self.unconfirmed_transactions:
            return False

        last_block = self.last_block
```

SCREEN SHOTS

To run project double click on ‘run.bat’ file to get below screen


Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Providers
Farmers
Processors
Distributors
Retailers
Consumers
Save ASC Details
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

In above screen different users can add their product details and each details will be stored in Blockchain and in above screen enter some details


Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

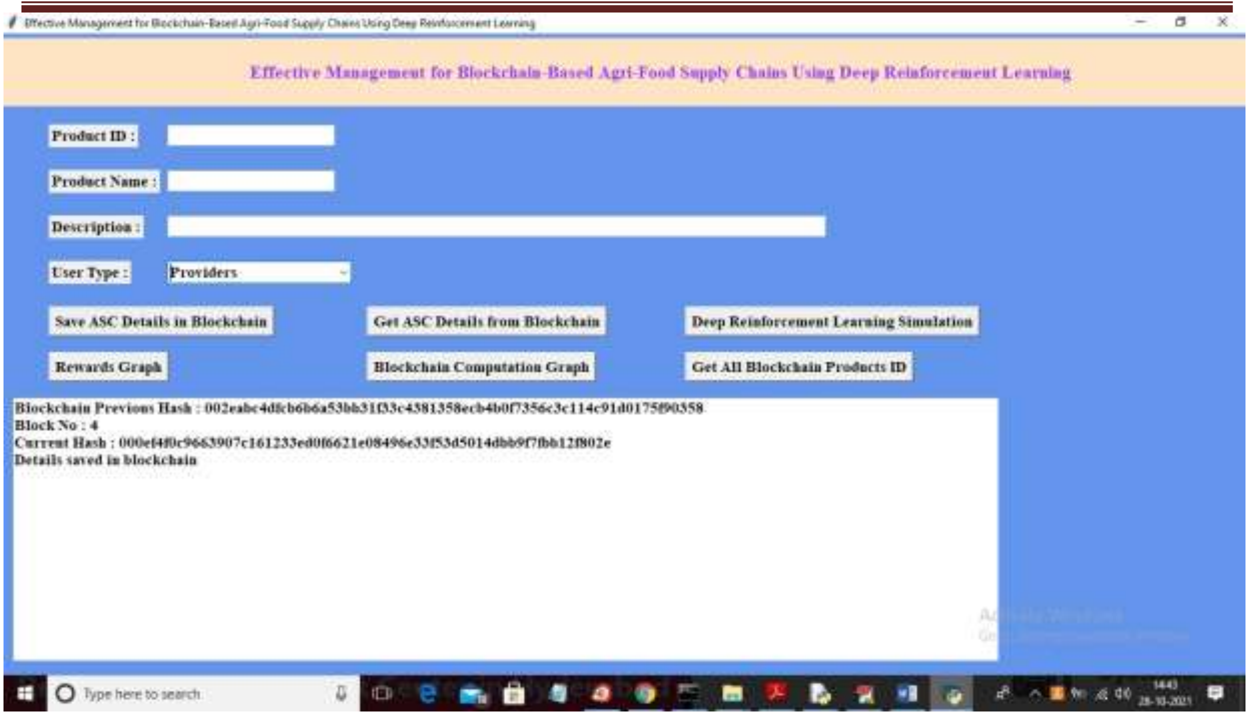
Product ID : Jowar

Product Name : Jowar Seed

Description : Telangana Seed manufacturer and providers (seeds with A1 quality)

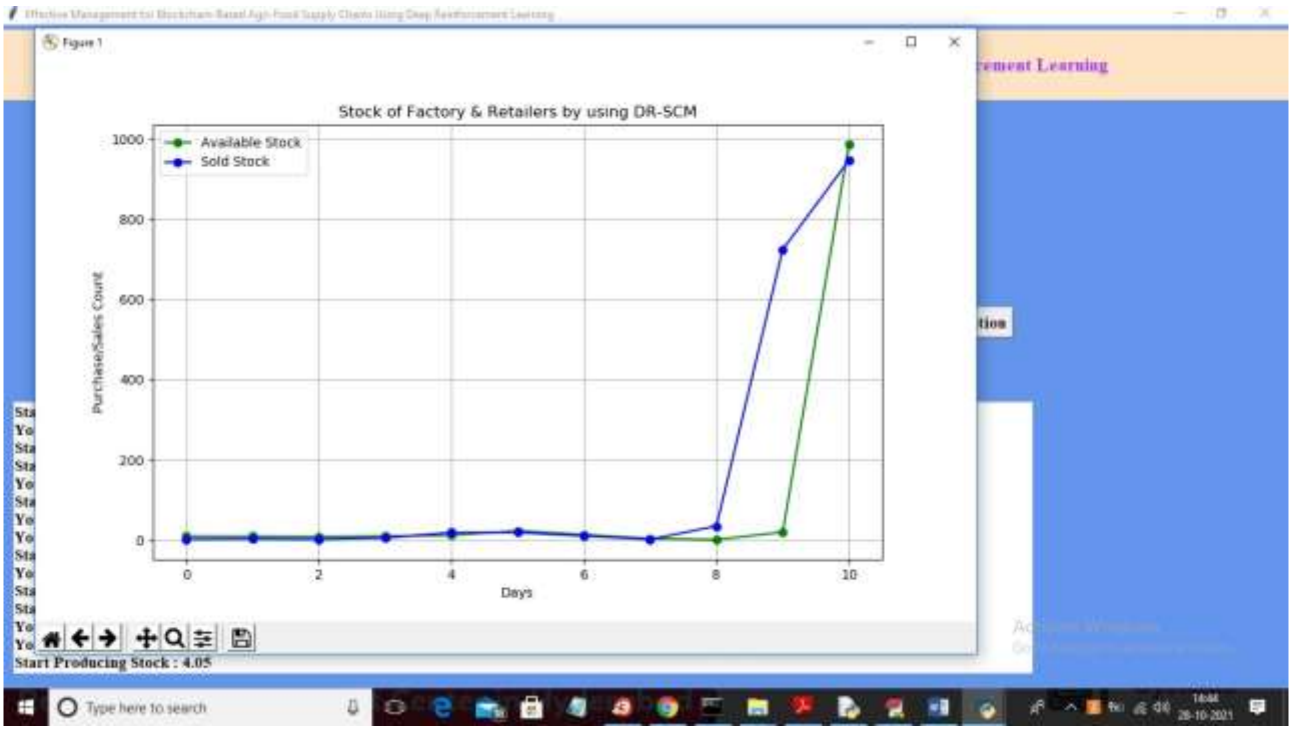
User Type : Providers

In above screen I entered some product details and then select user type and press ‘Save ASC Details in Blockchain’ button to store data in Blockchain.

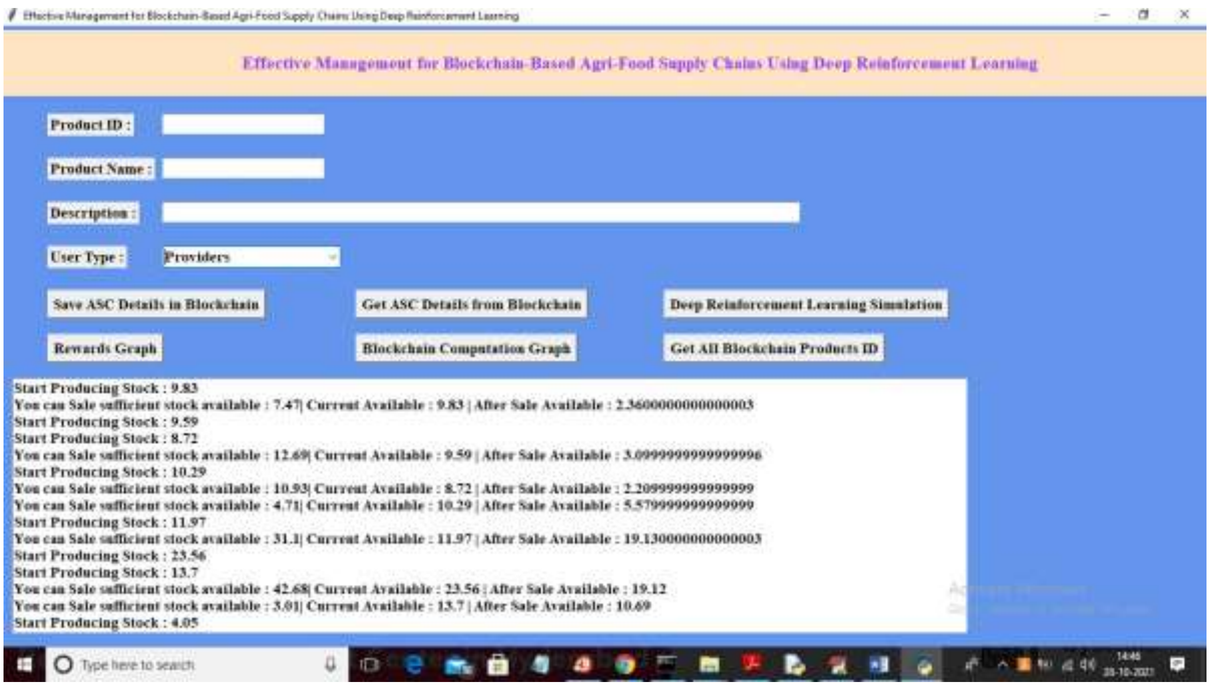

Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID
Type here to search

```
Blockchain Previous Hash : 002eabc4dfcb6b6a53bb31f33c4381358ecb4b0f7356c3c114c91d0175f90358
Block No : 4
Current Hash : 000ef4f0c9663907c161233ed0f6621e08496e33f53d5014dbb9f7fbb12f802e
Details saved in blockchain
```

In above screen we can see Hash codes generated for current transaction and we got old block hash code also with current block no. Similarly any user can add their product details and now click on 'Deep Reinforcement Learning Simulation' button to predict stock quantity

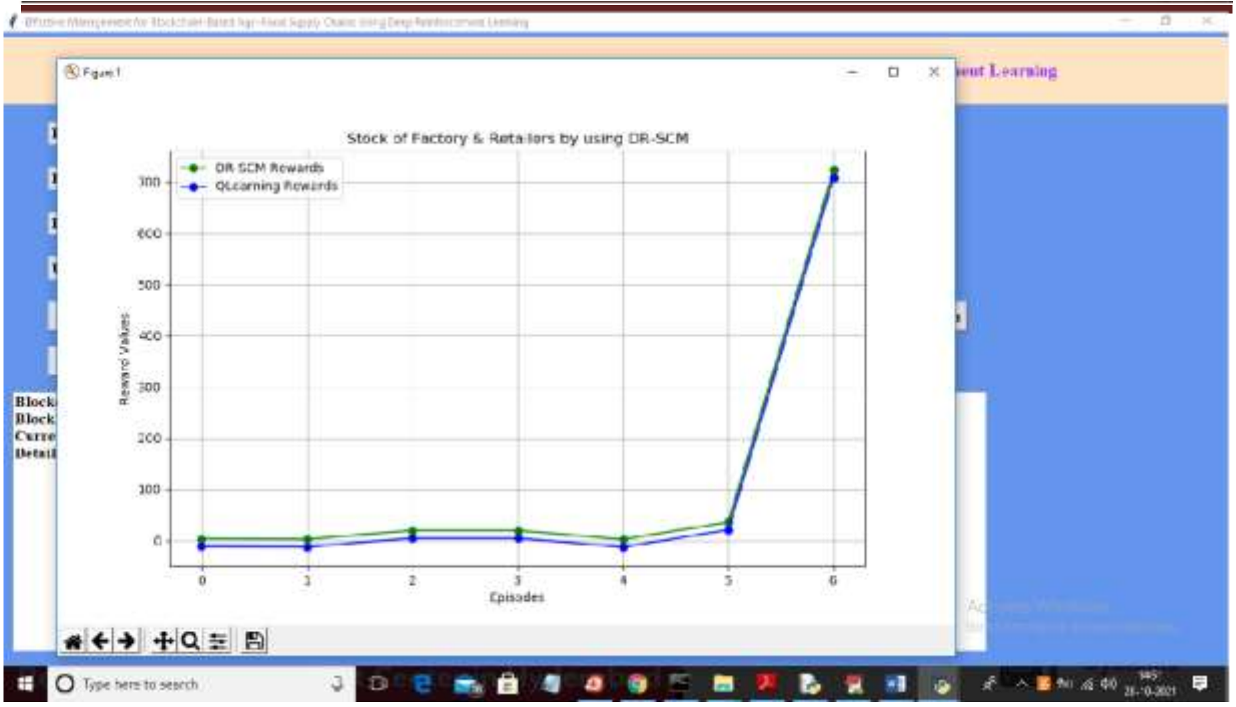


In above graph we can see DR-SCM stock prediction where green line represents available stock and blue line represents sold stock and based on above stock graph Framer will go for stock manufacturing and below screen will give complete details on prediction. In above graph x-axis represents number of days and y-axis represents stock quantity

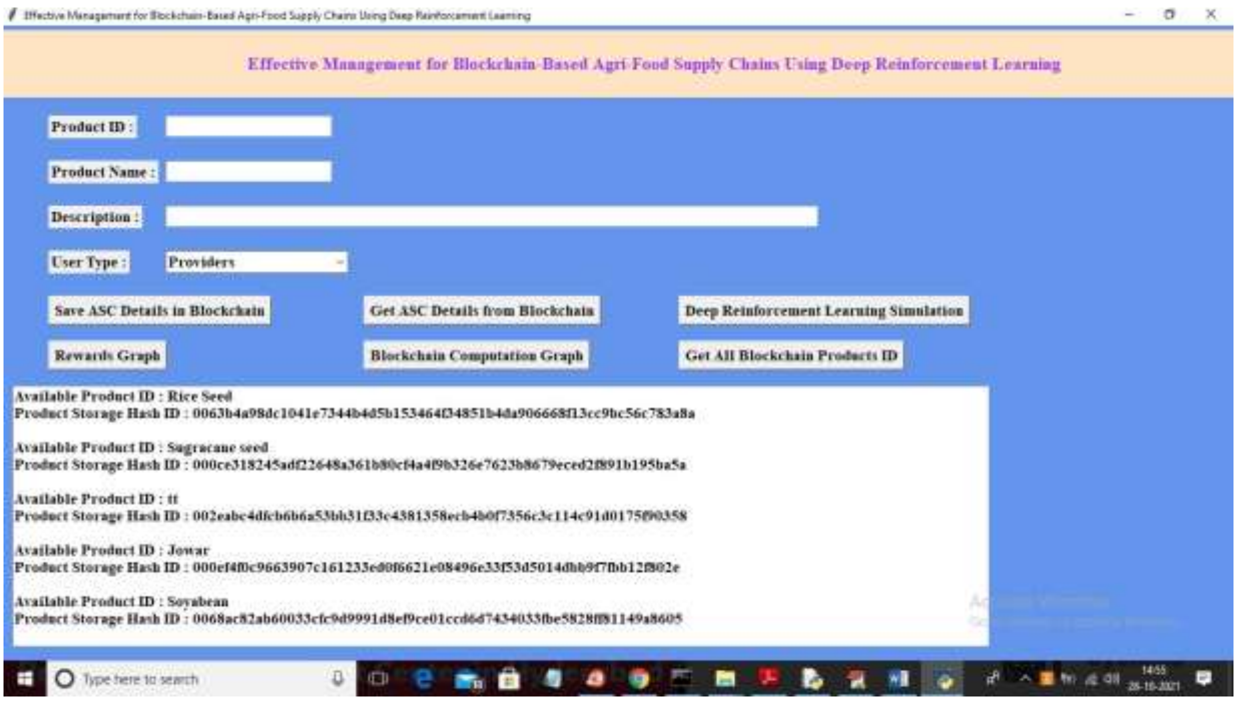

Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

```
Start Producing Stock : 9.83
You can Sale sufficient stock available : 7.47| Current Available : 9.83 | After Sale Available : 2.3600000000000003
Start Producing Stock : 9.59
Start Producing Stock : 8.72
You can Sale sufficient stock available : 12.69| Current Available : 9.59 | After Sale Available : 3.0999999999999996
Start Producing Stock : 10.29
You can Sale sufficient stock available : 10.93| Current Available : 8.72 | After Sale Available : 2.2099999999999999
You can Sale sufficient stock available : 4.71| Current Available : 10.29 | After Sale Available : 5.5799999999999999
Start Producing Stock : 11.97
You can Sale sufficient stock available : 31.1| Current Available : 11.97 | After Sale Available : 19.130000000000003
Start Producing Stock : 23.56
Start Producing Stock : 13.7
You can Sale sufficient stock available : 42.68| Current Available : 23.56 | After Sale Available : 19.12
You can Sale sufficient stock available : 3.01| Current Available : 13.7 | After Sale Available : 10.69
Start Producing Stock : 4.05
```

In above screen for next 10 days DRL has given prediction for ‘start producing stock’ or ‘sale stock’ and if available stock less than sale stock then it’s wrong prediction and in above screen we can see more number of times algorithm has predicted correctly. Now click on ‘Rewards Graph’ to get below graph.



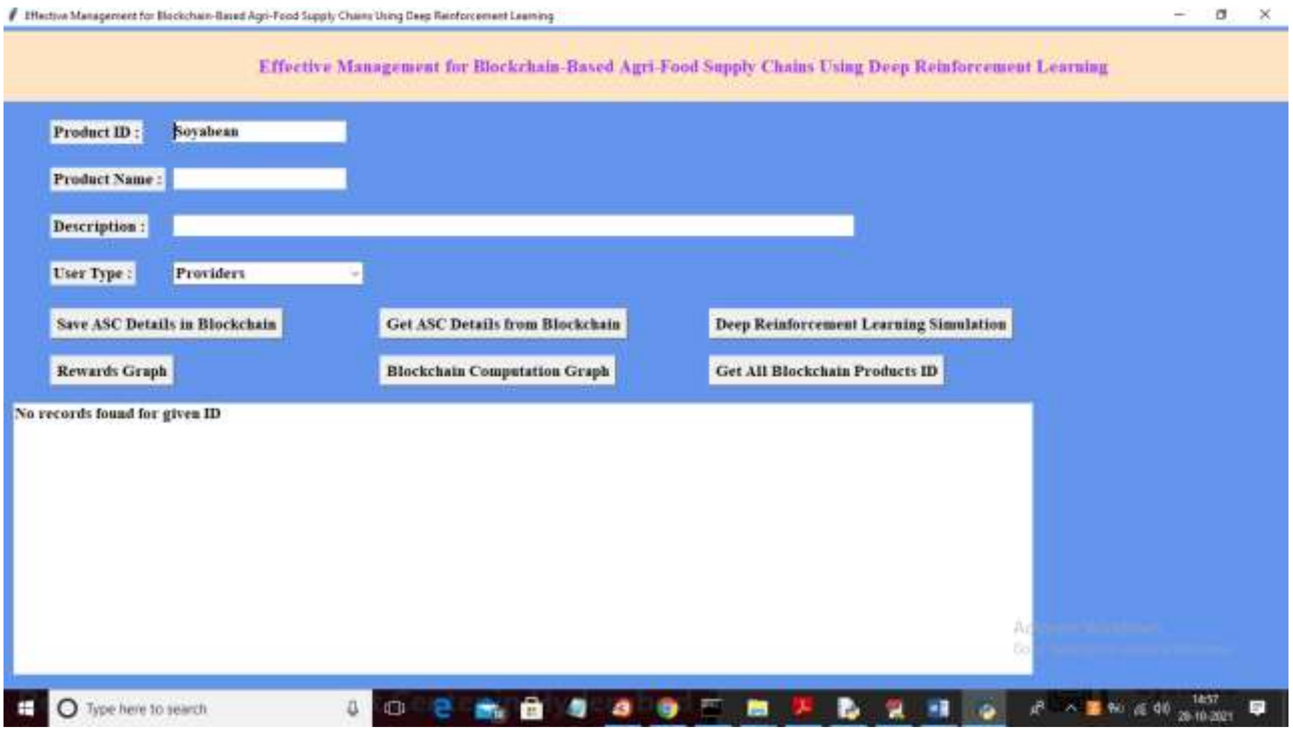
In above graph we can see DR-SCM and QLearning rewards where x-axis represents EPISODES/iteration and y-axis represents rewards and if algorithm predict correctly then its reward values will increase else decrease. In above graph green line represents DR-SCM rewards and blue line represents QLearning and in above graph we can see DR-SCM got more rewards values. Now click on ‘Blockchain Computation Graph’ button to get below Blockchain hash computation graph


Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID
Type here to search

```
Available Product ID : Rice Seed
Product Storage Hash ID : 0063b4a98dc1041e7344b4d5b153464f34851b4da906668f13cc9bc56c783a8a

Available Product ID : Sugracane seed
Product Storage Hash ID : 000ce318245adf22648a361b80cf4a4f9b326e7623b8679eced2f891b195ba5a

Available Product ID : tt
Product Storage Hash ID : 002eabc4dfcb6b6a53bb31f33c4381358ecb4b0f7356c3c114c91d0175f90358

Available Product ID : Jowar
Product Storage Hash ID : 000ef4f0c9663907c161233ed0f6621e08496e33f53d5014dbb9f7fbb12f802e

Available Product ID : Soyabean
Product Storage Hash ID : 0068ac82ab60033cfc9d9991d8ef9ce01ccd6d7434033fbe5828ff81149a8605
```

In above screen we can see names of each product ID and its block hash code and now enter any product id in first text field and then click on ‘Get ASC Details from Blockchain’ button to get all details of that product

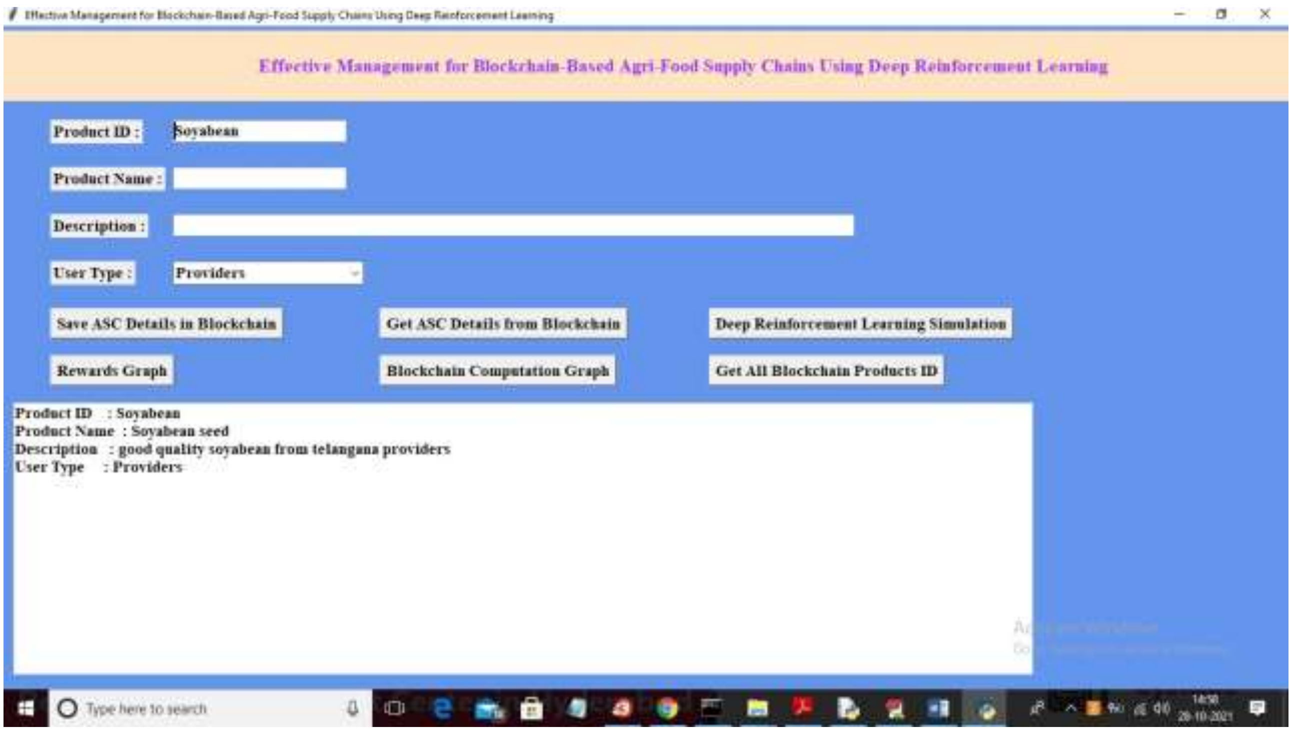
ID


Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

Product ID : Soyabean
User Type : Providers

```
No records found for given ID
```

In above screen I entered some product id in text field and now click on 'Get ASC Details from Blockchain' button to get below output

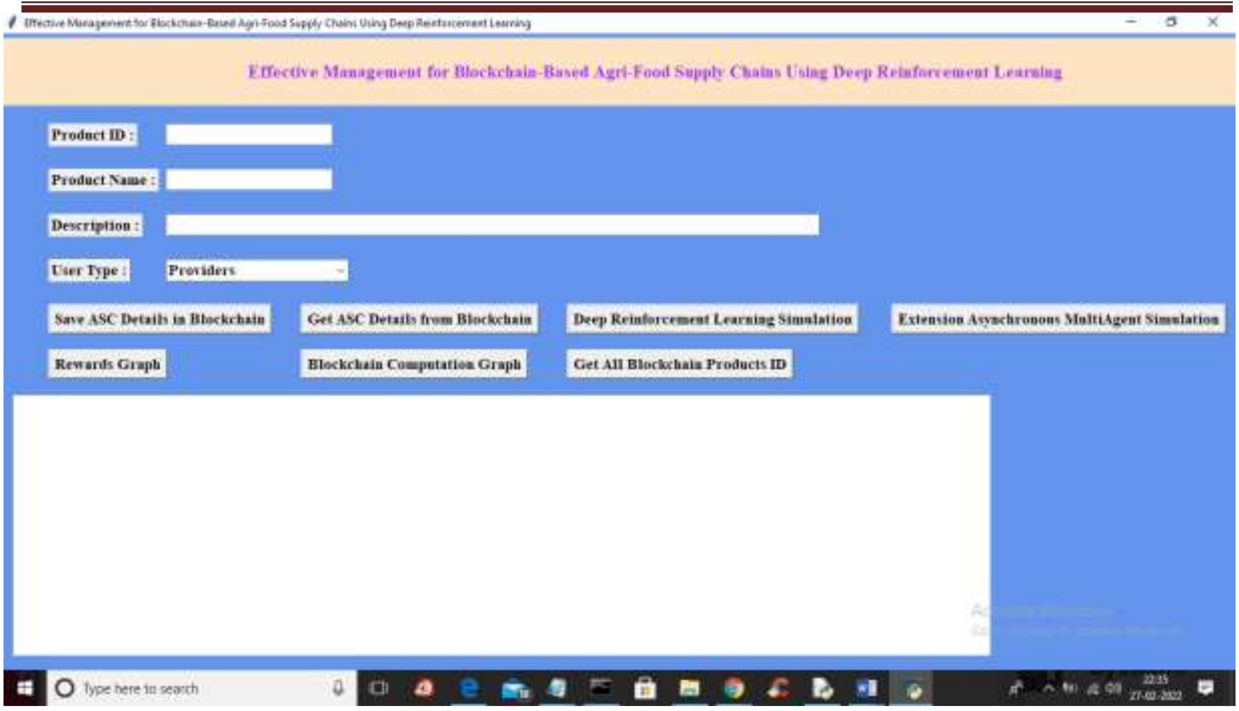

Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

Product ID : Soyabean
User Type : Providers

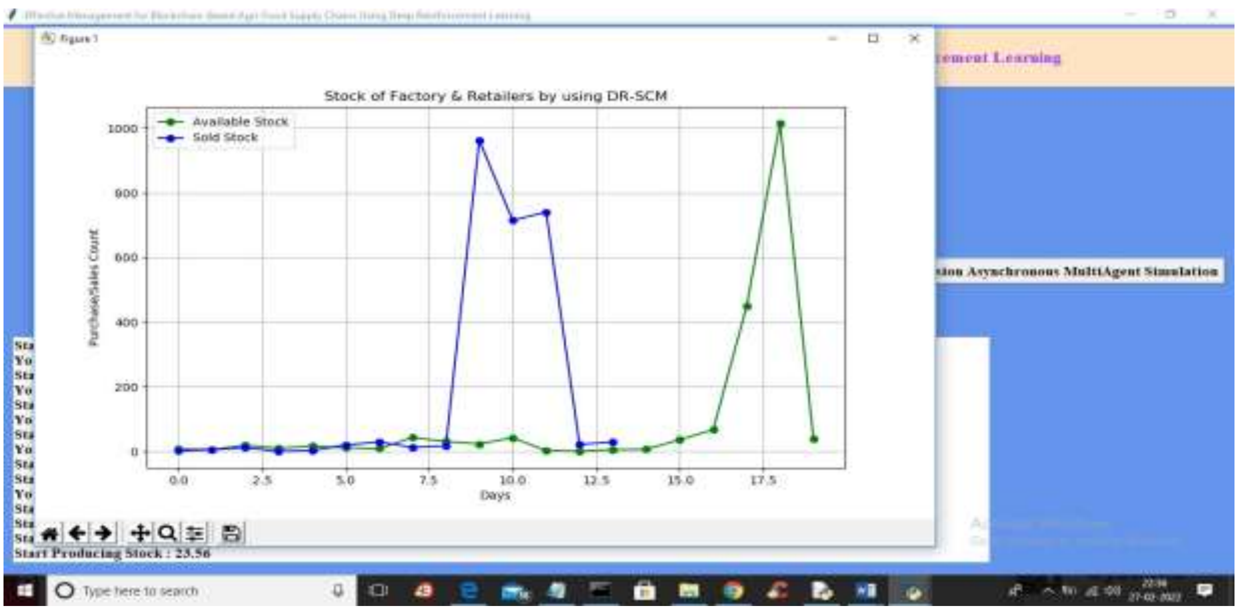
```
Product ID    : Soyabean
Product Name  : Soyabean seed
Description   : good quality soyabean from telangana providers
User Type     : Providers
```

In above screen for given product ID we got details from Blockchain and similarly you can get details for all products you stored

To run project double click on 'run.bat' file to get below screen

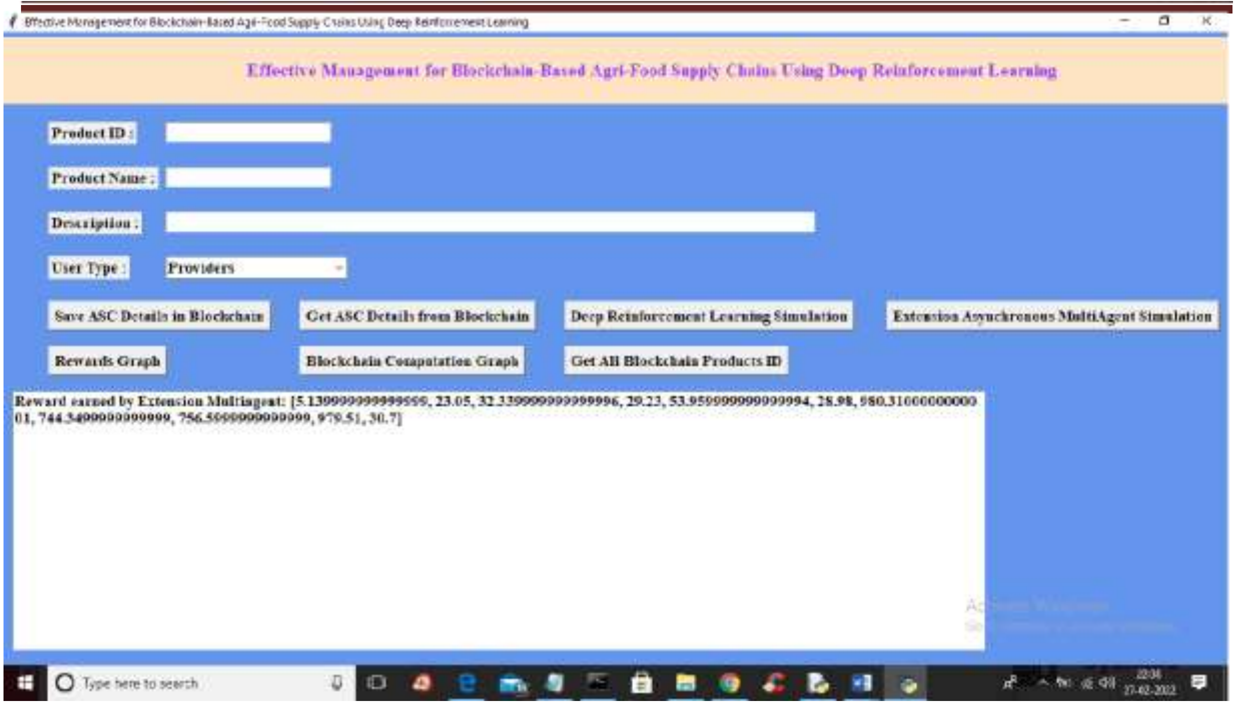

Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Extension Asynchronous MultiAgent Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID
Type here to search
22:35
27-02-2022

In above screen to run extension concept first double click on ‘Deep Reinforcement Learning Simulation’ button to train model with single agent and get below output



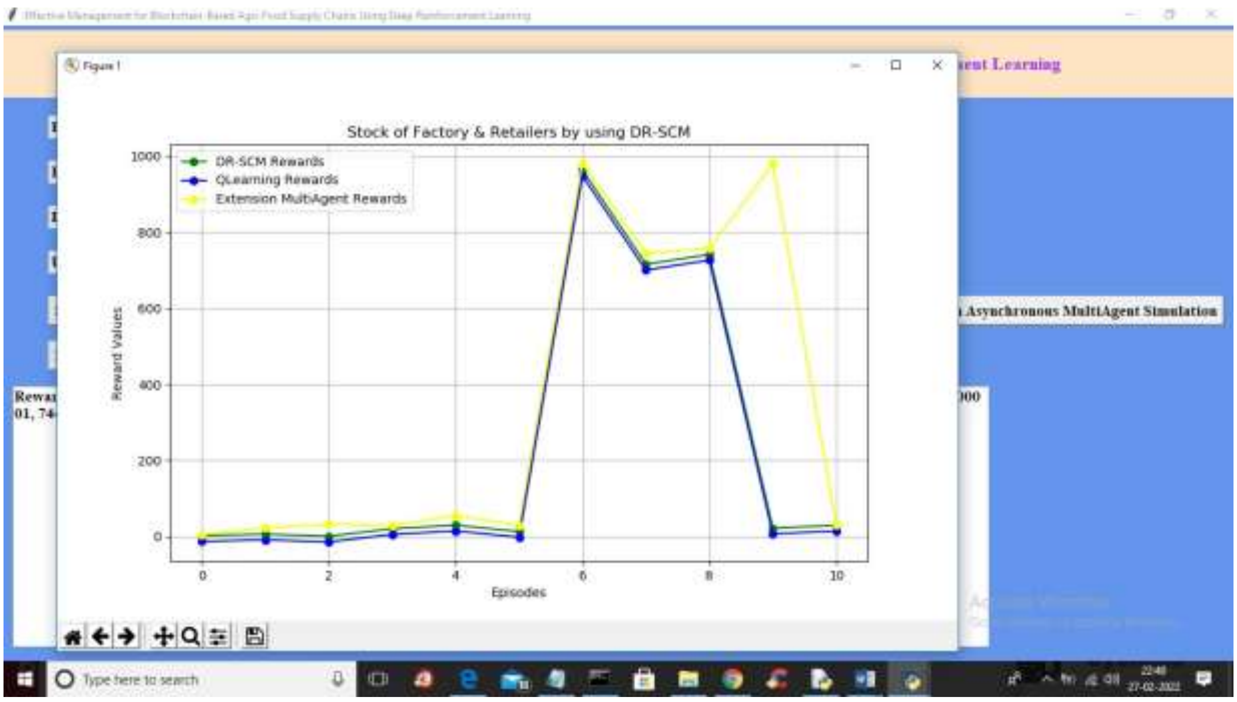
Start Producing Stock : 23.56

In above screen we got sold and available stock details with single agent and now click on ‘Extension Asynchronous MultiAgent Simulation’ button to train model with multi agent and get below output


Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning
Product ID :
Product Name :
Description :
User Type :
Providers
Save ASC Details in Blockchain
Get ASC Details from Blockchain
Deep Reinforcement Learning Simulation
Extension Asynchronous MultiAgent Simulation
Rewards Graph
Blockchain Computation Graph
Get All Blockchain Products ID

```
Reward earned by Extension Multiagent: [5.139999999999999, 23.05, 32.339999999999996, 29.23, 53.959999999999994, 28.98, 980.3100000000001, 744.3400000000009, 756.5000000000009, 979.51, 30.7]
```

In above screen we can see rewards earned by multi agent to get optimize performance and to get this output it may take some seconds of time as multi agent will keep learning from model to get better performance and now click on 'Rewards Graph' button to get below output



In above graph blue line represents existing QLearning model and green line represents propose DR-SCM algorithm and yellow line represents multi agent learning and in all algorithms extension multi agent got more rewards. In above graph x-axis represents training episodes/epoch and y-axis represents total earned by These algorithms.

CHAPTER-12

CONCLUSION AND FUTURE ENHANCEMENT

12. CONCLUSION AND FUTURE ENHANCEMENT

The aim of this project to design a blockchain-based framework to guarantee the agri-food safety with product traceability in ASC systems. Next, we propose a DR-SCM method to make decisions on the production and storage of agri-food products for optimizing product profits in ASCs. The extensive simulation experiments verify the effectiveness of the proposed blockchain-based framework and the DR-SCM method for ASC optimization. More specifically, the results show that the proposed blockchain-based ASC framework can well guarantee reliable product traceability. Moreover, the DR-SCM outperforms common heuristic and Q-learning methods in terms of rewards (i.e. product profits) while achieving high learning efficiency in different scenarios of ASC management. Meanwhile, the DR-SCM has higher flexibility than others in arranging production and storage. In the real-world ASC environment, the demands from consumers are changing during different time periods. Based on the simulation experiments conducted by using the DR-SCM, the macro-control for the production and storage of agricultural products can be effectively performed. Thus, according to demands and costs, the production of factories can maintain available for retailers in a cost-effective way while the stock of retailers can well satisfy the demands from consumers. The DQN algorithm utilizes a mechanism of experience replay to facilitate convergence, but the experience data in the playback memory reveals strong relevance, which may cause the low efficiency of training for achieving the optimal performance.

FUTURE WORK

To address this problem, in the future, we will continue our research by applying other advanced DRL-based algorithms (e.g. asynchronous advantage actor-critic) in more complex scenarios of ASC management with the demands constructed by using real-world data. Meanwhile, we will evaluate the robustness and potential improvements by using these algorithms and explore their feasibility in real-world ASC environments.

CHAPTER-13
BIBILOGRAPHY

13.BIBILOGRAPHY

[1] D.-Y. Lin, C.-J. Juan, and C.-C. Chang, ‘‘Managing food safety with pricing, contracts and coordination in supply chains,’’ IEEE Access, vol. 7, pp. 150892–150909, 2019.

[2] H. Fan, ‘‘Theoretical basis and system establishment of China food safety intelligent supervision in the perspective of Internet of Things,’’ IEEE Access, vol. 7, pp. 71686–71695, 2019.

[3] M. Toledo-Hernández, T. Tscharntke, A. Tjoa, A. Anshary, B. Cyio, and T. C. Wanger, ‘‘Hand pollination, not pesticides or fertilizers, increases cocoa yields and farmer income,’’ Agricult., Ecosyst. Environ., vol. 304, Dec. 2020, Art. no. 107160.

[4] J. Himmelstein, A. Ares, D. Gallagher, and J. Myers, ‘‘A meta-analysis of intercropping in Africa: Impacts on crop yield, farmer income, and integrated pest management effects,’’ Int. J. Agricult. Sustainability, vol. 15, no. 1, pp. 1–10, Jan. 2017.

[5] Y. Dong, Z. Fu, S. Stankovski, S. Wang, and X. Li, ‘‘Nutritional quality and safety traceability system for China’s leafy vegetable supply chain based on fault tree analysis and QR code,’’ IEEE Access, vol. 8, pp. 161261–161275, 2020.

[6] C. Ganeshkumar, M. Pachayappan, and G. Madanmohan, ‘‘Agri-food supply chain management: Literature review,’’ Intell. Inf. Manage., vol. 9, no. 2, pp. 68–96, 2017.

[7] Q. Lin, H. Wang, X. Pei, and J. Wang, ‘‘Food safety traceability system based on blockchain and EPCIS,’’ IEEE Access, vol. 7, pp. 20698–20707, 2019.

[8] H. Feng, X. Wang, Y. Duan, J. Zhang, and X. Zhang, ‘‘Applying blockchain technology to improve agri-food traceability: A review of development methods, benefits and challenges,’’ J. Cleaner Prod., vol. 260, Jul. 2020, Art. no. 121031.

[9] S. Nakamoto, ‘‘Bitcoin: A peer-to-peer electronic cash system,’’ White Paper, 2008. Accessed: Jun. 26, 2020. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[10] F. Tian, ‘‘An agri-food supply chain traceability system for China based on RFID & blockchain technology,’’ in Proc. 13th Int. Conf. Service Syst. Service Manage. (ICSSSM), Jun. 2016, pp. 1–6.