# BLOCKCHAIN BASED E-COMMERCE ONLINE APPLICATION

#### A MINI PROJECT REPORT

***Submitted by***

**MOHAMMED KHALEEL UDDIN SIDDIQUI**

**(20841A1211)**

**P. MADHU**

**(21845A1202)**

**KOLEPAKA SAMPATH RAJU**

**(20841A1223)**

**GUIDED BY**

**Mr. A. R. SOFI**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

### INFORMATION TECHNOLOGY



#### AURORA’S TECHNOLOGICAL AND RESEARCH INSTITUTE

###### **(Affiliated to JNTU, Hyderabad and Accredited by NAAC with ‘A’ grade)**

###### Parvathapur, Uppal, Hyderabad-500 039

###### **DECEMBER 2023**

#### AURORA’S TECHNOLOGICAL AND RESEARCH INSTITUTE

###### **(Affiliated to JNTU, Hyderabad and Accredited by NAAC with ‘A’ grade)**

Parvathapur, Uppal, Hyderabad-500 039



###### **DECLARATION**

We hereby declare that the work described in this project, entitled  **‘BLOCKCHAIN BASED E-COMMERC ONLINE APPLICATION’** which is being submitted by us in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to **AURORA’S TECHNOLOGICAL AND RESEARCH INSTITUTE** is the result of investigation carried by us under the guidance of **Mr. A. R. Sofi, Associate Professor, Department of CSE/IT.**

The work is original and has not been submitted for any degree of this or any other university.

Place: Hyderabad

Date:

**MOHAMMED KHALEEL UDDIN SIDDIQUI (20841A1211)**

**P.MADHU (21845A1202)**

**KOLEPAKA SAMPATH RAJU (20841A1223)**

**CERTIFICATE**

Certified that this project report **BLOCKCHAIN BASED E-COMMERCE ONLINE APPLICATION** is the bonafide work of **MOHAMMED KHALEEL UDDIN SIDDIQUI (20841A1211)**, **P.MADHU (218451202), K SAMAPTH RAJU (20841A1223)** who carried out the project work under our supervision.

|  |  |
| --- | --- |
| **GUIDE** | **MINI PROJECT COORDINATOR** |
| **Mr. A. R. SOFI** | **Mrs. D. SUBHASHINI** |
| Associate Professor | Associate Professor |
|  |  |
|  |  |
| **HEAD OF THE DEPARTMENT** | **PRINCIPAL** |
| **Mrs. K. KAVITHA** | **Dr. A. MAHESH BABU** |
|  |  |
|  |  |

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

This work has been done during the project period and it was a very good opportunity to put theoretical knowledge into planned exercise with an aim to solve a real time problem and also to develop confidence to face various practical situations.

We convey thanks to our project guide **Mr. A.R. Sofi,** Department of Computer Science and Engineering, for providing encouragement, constant support and guidance which was of great help to complete this project successfully.

We express our sincere thanks to Project Coordinator **Mrs. D. Subhashini** for helping us to complete our project work by giving valuable suggestions.

We express our sincere thanks to Head of the Department **Mrs. K. Kavitha,** for giving us the support and her kind attention and valuable guidance to us throughout this course.

We would also like to express our gratitude to **Dr. A. Mahesh Babu, Principal** Aurora’s Technological and Research Institute for providing us with a congenial atmosphere and encouragement.

**ABSTRACT**

In existing e-commerce application all customers and product details will be stored and managed in single centralized server and if this server crashed due to too many requests and or if server is hacked then services will not be available to other customers and to overcome from this problem we are migrating e-commerce application to Blockchain which will maintain data at multiple nodes/servers and if one node down then customers can get data from other working nodes. Another advantage of Blockchain has inbuilt support for data encryption and immutable (data cannot be alter by unauthorized users) and it will consider each data as block/transaction and associate each block storage with unique hash code and before storing new records Blockchain will verify hash code of previous blocks and if all nodes blocks verification successful then data is consider as secured.

To implement this project we have used Blockchain Ethereum with Truffle Store e-commerce data and Blockchain cannot store images so we are storing products images inside IPFS (interplanetary file storage) server and this server will store image and returned hash code of stored image and by giving that hash code we can retrieve images from IPFS.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ACKNOWLEDGEMENT** | iv |
|  | **ABSTRACT** | v |
|  | **LIST OF FIGURES** | viii |
|  | **LIST OF TABLES** | ix |
|  | **LIST OF ABBREVIATIONS** | x |
| **1.** | **INTRODUCTION** | 1 |
|  | * 1. Overview   2. Scope   3. Definitions And Acronyms   4. User Needs   5. Assumptions | 1  1  2  2  2 |
| **2.** | **LITERATURE SURVEY** | 3 |
| **3.** | **SYSTEM STUDY** | 7 |
|  | 3.1 Problem Statement | 7 |
|  | 3.2 Existing System | 7 |
|  | 3.3 Proposed System | 8 |
|  | 3.4 Feasibility Study | 8 |
| **4.** | **SOFTWARE REQUIREMENTS ANALYSIS** | 10 |
|  | 4.1 Methodology | 10 |
|  | 4.3 Functional Requirements | 14 |
|  | * 1. Non-Functional Requirements | 15 |
| **5.** | **REQUIREMENT SPECIFICATION** | 17 |
|  | 5.1 Hardware Requirements | 17 |
|  | 5.2 Software Requirements | 17 |
| **6.** | **SOFTWARE DESIGN** | 18 |
|  | * 1. Architectural Design | 18 |
|  | * 1. Modules | 18 |
|  | * 1. Uml Diagrams | 20 |
| **7.** | **SOURCE CODE** | 25 |
| **8.** | **SYSTEM TESTING** | 33 |
|  | 8.1 Test Cases | 33 |
| **9.** | **OUTPUT SCREENS** | 38 |
| **10.** | **CONCLUSION** | 46 |
| **11.** | **REFERENCES** | 47 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
| 6.1 | System Architecture | 18 |
| 6.2 | Use-Case Diagram | 21 |
| 6.3 | Sequence Diagram | 22 |
| 6.4 | Class Diagram | 23 |
| 6.5 | Activity Diagram | 24 |
| 9.1 | Users Signup | 38 |
| 9.2 | Supplier Signup | 38 |
| 9.3 | Supplier Signup Completed | 39 |
| 9.4 | Supplier Login | 39 |
| 9.5 | Add Products | 40 |
| 9.6 | New Products | 40 |
| 9.7 | Products Added | 41 |
| 9.8 | Update Product Quantity | 41 |
| 9.9 | Quantity Updated | 42 |
| 9.10 | Customer Login | 42 |
| 9.11 | Browse Products | 43 |
| 9.12 | Search Products | 43 |
| 9.13 | View Products | 44 |
| 9.14 | Place Order | 44 |
| 9.15 | View Products | 45 |
| 9.16 | View Orders Details | 45 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| 9.1 | Test case 1 Supplier Signup | 33 |
| 9.2 | Test case 2 Customer Signup | 33 |
| 9.3 | Test case 3 Supplier Login | 34 |
| 9.4 | Test case 4 Customer Login | 34 |
| 9.5 | Test case 5 Invalid Login | 35 |
| 9.6 | Test case 6 Add Product | 35 |
| 9.7 | Test case 7 Update Quantity | 36 |
| 9.8 | Test case 8 View Orders | 36 |
| 9.9 | Test case 9 Browse Product | 37 |
| 9.10 | Test case 10 Browse Product | 37 |

**LIST OF ABBREVIATIONS**

RAM - Random Access Memory

IPFS - Interplanetary File System

POW - Proof of Work

ETH - Ethereum

POA - Proof of Accomplishment

CWI - Centrum Wiskunde &Informatica

DAG - Directed Acyclic Graph

**CHAPTER 1**

**INTRODUCTION**

The development of a blockchain based e-commerce online application has the potential to revolutionize the way people shop and do business online. By leveraging the key features of blockchain technology like decentralization, transparency, security and automation through smart contracts, such an application could solve many of the challenges faced by traditional e-commerce platforms. Centralized databases and payment gateways leave existing e-commerce platforms vulnerable to data breaches, fraud, payment disputes and supply chain inefficiencies. A blockchain based solution could provide a single, immutable record of all transactions that is distributed across a network of nodes, ensuring data integrity and security.

**1.1 OVERVIEW**

The proposed Blockchain-based e-commerce online application aims to address the limitations of traditional centralized e-commerce systems by leveraging Blockchain technology. In the existing system, all customer and product data is stored on a single centralized server, leading to potential vulnerabilities such as crashes due to high traffic or hacking incidents. The proposed solution employs a decentralized approach using ETH Blockchain with Truffle for data storage and integrity verification. Additionally, images of products are stored on an IPFS (Inter Planetary File System) server, enhancing data security.

**1.2 SCOPE**

The scope of this project includes the development of a secure e-commerce platform that utilizes Blockchain for data storage and retrieval. The application will feature modules for user authentication, product management, order processing, and browsing products. By decentralizing data across multiple nodes, the system aims to ensure uninterrupted service availability even if some nodes experience issues. By integrating blockchain, we aim to enhance transparency and trust throughout the e-commerce experience. Users will benefit from enhanced data ownership and immutability, while businesses can operate with increased security and streamlined transaction processing.

**1.3 DEFINITIONS AND ACRONYMS:**

* **E-Commerce**: Electronic commerce, the buying and selling of goods and services over the internet.
* **Blockchain:** A decentralized and distributed digital ledger that records transactions across multiple computers in a secure and transparent manner.
* **Ipfs:** Interplanetary file system, a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system.
* **Truffle:** A development environment, testing framework, and asset pipeline for ethereum.

**1.4 USER NEEDS**

* **Customers:**

Secure and private transactions.

Easy product search and ordering functionality.

Access to accurate and up-to-date product information.

* **Suppliers:**

Secure product data management on the Blockchain.

Real-time updates on product quantities.

Efficient order processing and management.

**1.5 ASSUMPTIONS AND DEPENDENCIES**

Several key assumptions and dependencies underpin the successful development and operation of this platform. Firstly, users are assumed to possess basic knowledge of e-commerce operations, ensuring a smooth user experience. Additionally, reliable internet connectivity is essential for seamless interaction with the platform, particularly for blockchain-based transactions. Furthermore, the platform's functionality hinges on the continued availability and performance of the Ethereum Blockchain and IPFS services .On the technical side, Python forms the foundation for the platform's development, requiring developers to be familiar with its syntax and features. The operating system must be Windows 7 or a later version, and the Chrome browser is needed for crucial debugging and emulation tasks during development.

**CHAPTER 2**

**LITERATURE REVIEW**

In their paper titled "Efficient and Secure Consensus and Privacy Protection Scheme for E-Commerce Consortium Blockchain," authors Haoyang Wu, Guangshun Li, and Junhua Wu propose a comprehensive solution to address critical challenges in e-commerce consortium blockchains. The research is driven by the shortcomings of existing solutions, specifically related to scalability, privacy protection, and vulnerability to attacks within consortium blockchains designed for e-commerce applications. The authors highlight the advantages of consortium blockchains over public and private alternatives in the context of e-commerce but emphasize the need for optimization [1]. The theoretical framework draws upon a combination of distributed ledger technology, cryptography, consensus algorithms, performance modeling, and security analysis to evaluate proposed improvements tailored for the e-commerce domain [1].

Identifying key research gaps, the paper underscores the deficiencies in current e-commerce-focused consortium blockchains, pointing out issues related to performance, privacy preservation, security, and decentralization. Additionally, the absence of interoperability standards is recognized as a hindrance. The methodology adopted involves a comprehensive literature review followed by the proposition and implementation of a novel framework [1].

The paper titled "Revolutionizing E-Commerce Using Blockchain Technology and Implementing Smart Contracts," authors Mohammad Monirujjaman Khan, Tasneem RoJa, Faris A. Almalki, and Maha Aljohani introduce a novel system that combines blockchain and smart contracts to address critical issues related to data privacy and security in the realm of e-commerce [2]. The integration of blockchain provides a decentralized and immutable infrastructure for data storage and transfer, enhancing security and eliminating vulnerabilities associated with centralized systems and digital transactions. Smart contracts, a key component of the proposed system, automate and secure transactions programmatically, leading to error-free e-commerce processes. The authors present results indicating the effectiveness of this approach, emphasizing its potential for establishing secure and trustworthy e-commerce transactions [2]. The background and context of the paper highlight the inherent security risks in digital transactions and centralized systems within the e-commerce landscape. Blockchain, by offering decentralized and transparent transaction ledgers, is positioned as a solution to enhance security, reduce fraud, and eliminate intermediaries. However, challenges related to scaling, governance, and regulatory issues are acknowledged. The theoretical framework underscores key aspects of blockchain, such as decentralization, immutability, transparency, and cryptography, which collectively contribute to secure and trusted data exchange [2].The authors identify existing research gaps, emphasizing persisting issues in privacy, scalability, integration, and security risks within blockchain-based e-commerce [2]. The methodology involves the use of JavaScript for blockchain, Solidity for smart contracts, and web interfaces with HTML/CSS/PHP to build a system that showcases the efficiency, privacy, and trustlessness of e-commerce transactions.

The transformative potential of blockchain technology in revolutionizing e-commerce accessibility, particularly focusing on diverse products such as health medicines, electronics, securityappliances, and food. The proposed system, aptly named 'PRODCHAIN,' employs a generic blockchain framework incorporating lattice-based cryptographic processes to simplify product tracing within the e-commerce landscape. Introducing the Proof of Accomplishment (PoA) consensus process adds a novel dimension to the solution. Experimental studies conducted on the Ethereum network underscore the system's efficiency in terms of latency and throughput, emphasizing its capability to enhance e-commerce products and services [3]. The backdrop of the paper underscores the limitations of current centralized e-commerce models, pointing out vulnerabilities like a single point of failure, lack of transparency, and security risks. Blockchain, as a decentralized alternative, is positioned as a solution to address these issues, providing a transparent, secure, and resilient network for e-commerce transactions. The theoretical framework delves into distributed ledger technology as the core model enabling blockchain-based e-commerce, citing agency theory and transaction cost economics to explain its potential impact on reducing information asymmetry and transaction costs. The paper identifies research gaps, emphasizing the need for further exploration of challenges related to scalability, governance, regulatory issues, adoption barriers, integration with existing systems, and user experience in blockchain-based e-commerce. The methodology involves the use of JavaScript for blockchain, Solidity for smart contracts [3] and web interfaces with HTML/CSS/PHP to build a system that showcases the efficiency, privacy, and trust lessness of e-commerce transactions.[3] The primary themes and trends center around decentralization through blockchain technology and the overarching goal of improving accessibility for disabled users.

"Research on Real-Time Tracking Algorithm of E-Commerce Logistics Information Based on Blockchain Technology" by Yanjie Zhu [4] discusses how blockchain technology can improve data security in e-commerce. The paper proposes a real-time tracking algorithm for e-commerce logistics information based on blockchain to address issues of distrust in centralized information transfer modes. Blockchain's decentralized and transparent nature is highlighted as a solution to improve trust among various entities involved in e-commerce logistics. The proposed algorithm aims to provide an open and verifiable system for real-time logistics tracking, leveraging blockchain's features such as cryptography, consensus, and immutability of recorded data. By storing data on the chain only when validated by a majority of nodes, the model ensures that the data cannot be affected [4]. This approach enhances data security and trust in e-commerce logistics information sharing. The paper emphasizes the potential of blockchain to significantly improve the efficiency, transparency, and security of e-commerce logistics data

With the rapid advancement of cryptography and distributed computing systems, blockchain technologies are highly anticipated to transform many industries with better transparency, high security, and low transaction costs. However, the scalability and performance of blockchains are limiting their utility and suitability in online services, especially e-commerce [5].

A survey of blockchain technologies to highlight their benefits and challenges in online shopping. We, therefore, propose two blockchain-based e-commerce applications with detailed design guidelines: social shopping and loyalty program. The study contributes to the cumulative theoretical development of social computing and blockchains. It also provides a number of implications for academic bodies, platform operators, and developers of blockchain technologies [5].

Blockchain technology enhances supply chain management by enabling the tracking of products and components throughout the entire supply chain, from manufacturing to delivery. This facilitates swift issue resolution in quality control. Additionally, blockchain-based supply chain management reduces reliance on vulnerable paper records susceptible to theft or loss [6].The secure and tamper-proof nature of blockchain allows businesses to store transaction data, ensuring the verification of customer and supplier identities while tracking product movements and payments. This capability helps mitigate fraudulent activities and counterfeiting.

Blockchain systems are inherently transparent, fostering honest information sharing between customers and suppliers. Public records of customer ratings and reviews further contribute to building trust between businesses and their customers. Data traceability is a key advantage of blockchain, as all information is stored in a public ledger, enabling the tracking of data history [6]. However, the mention of cryptocurrency as the main mode of payment is omitted in this rephrasing.

**CHAPTER 3  
SYSTEM STUDY**

**4.1 PROBLEM STATEMENT**

In the realm of e-commerce, traditional systems grapple with a critical challenge – the vulnerability stemming from a centralized architecture. The centralized server, responsible for storing and managing all customer and product data, poses a substantial risk. This vulnerability manifests in two primary forms: the potential for server crashes due to an overwhelming influx of requests and the ever-looming threat of security breaches. In essence, the problem at hand revolves around the single point of failure inherent in centralized e-commerce applications. The consequence is the unavailability of services to users, a scenario that not only disrupts the user experience but also poses significant risks to data security and business continuity. The proposed solution seeks to transcend these limitations by migrating towards a Blockchain-based E-commerce application

**4.2 EXISTING SYSTEM**

In existing e-commerce application all customers and product details will be stored and managed in single centralized server and if this server crashed due to too many requests and or if server is hacked then services will not be available to other customers and to overcome from this problem we are migrating e-commerce application to Blockchain which will maintain data at multiple nodes/servers and if one node down then customers can get data from other working nodes.

**4.2.1 Disadvantages**

* Less Security
* Lack of automation
* Scalability issues
* Data vulnerabilities
* Counterfeit product
* No Transparency

**4.3 PROPOSED SYSTEM**

Blockchain offers inherent support for robust data encryption and immutability, ensuring that unauthorized users cannot alter stored information. In this project, we've leveraged Ethereum Blockchain with Truffle to securely store e-commerce data. Each piece of data is treated as a block or transaction, and each block is associated with a unique hash code. Before adding new records, Blockchain verifies the hash codes of previous blocks. If the verification is successful across all nodes, the data is considered secure and unaltered. However, as Blockchain itself cannot store images, we've employed IPFS (Inter-Planetary File Storage) to handle product images.

The IPFS server stores the images and returns a hash code, enabling the retrieval of images by providing the corresponding hash code. This integrated approach enhances the security and integrity of both textual and visual e-commerce data in our system.

**4.3.1 Advantages**

* High Security
* Enhanced Data Security
* Decentralization and Trust
* Transparency
  1. **FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**4.4.1 Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**4.4.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**4.4.3 Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CHAPTER 4**

**SOFTWARE REQUIREMENT ANALYSIS**

Software requirements analysis is a critical phase in the software development life cycle where analysts gather and document information about user needs and constraints. This involves techniques like interviews and surveys to elicit requirements, which are then documented clearly. The requirements are analyzed, prioritized, and verified for accuracy and completeness. Effective communication is key throughout the process.

.

**5.1 METHODOLOGY**

Blockchain is a decentralized and distributed digital ledger technology that enables the secure, transparent, and tamper-resistant recording of transactions across a network of computers. In essence, it consists of a chain of blocks, each containing a list of transactions, linked together in a chronological order. One of its key features is decentralization, meaning no single entity or authority has control over the entire network, enhancing trust and reducing the risk of manipulation.

Each block in the chain is cryptographically linked to the previous one through a hash function, creating a chain of blocks that cannot be altered retroactively. Consensus mechanisms, such as proof-of-work or proof-of-stake, ensure agreement among network participants on the validity of transactions. This technology is best known for underpinning cryptocurrencies like Bitcoin, but its applications extend beyond finance to various industries, providing solutions for secure and transparent record-keeping, smart contracts, and decentralized applications.

**Need for blockchain?**

The need for blockchain arises from the growing demand for a secure, transparent, and decentralized system for managing digital transactions and information. Traditional centralized systems face challenges such as vulnerabilities to hacking, data manipulation, and a lack of transparency. Blockchain addresses these issues by distributing data across a network of computers, eliminating a single point of control.

This technology has broad applications beyond cryptocurrencies, offering solutions for supply chain management, healthcare records, voting systems, and more. The need for blockchain is evident in industries seeking to streamline operations, enhance security, and establish a reliable foundation for trust in digital transactions, ultimately transforming the way we handle and verify information in the digital age.

**What makes blockchain technology so revolutionary?**

There are many potential benefits that come with the adoption of blockchain technology. Here are three to consider **Blockchain can drastically reduce data tampering.** Blockchain can significantly increase data security. This is why the technology is often called a “trust less network.”

**Blockchain can make transactions more transparent and traceable.** Because it’s a distributed ledger, all participating computers on a network have access to the same database (the blockchain itself). This increases transparency and access, and the hash history makes every exchange and transaction traceable.

**Challenges in Blockchain**

Despite the immense potential of blockchain technology, several challenges must be addressed for it to reach widespread adoption.

* **Scalability**

Blockchain networks, especially public ones, face challenges in scaling to accommodate a growing number of transactions. As the number of participants and transactions increases, scalability becomes a critical issue.

* **Interoperability**

Achieving interoperability between different blockchain platforms and networks is a challenge. Lack of standardized protocols hinders the seamless exchange of information and assets across diverse blockchain ecosystems.

* **Regulatory Uncertainty**

The regulatory landscape for blockchain and cryptocurrencies is still evolving. Varying regulations in different jurisdictions create uncertainty for businesses and users, affecting widespread adoption.

**Applications of Blockchain Technology**

Blockchain is being implemented in lager way to enhance it for further by building up high security.

* **Cryptocurrencies:**

Blockchain is the underlying technology for cryptocurrencies like Bitcoin, providing a decentralized and secure way to conduct digital transactions.

* **Smart Contracts**

Smart contracts are self-executing contracts with the terms directly written into code. They automate and enforce the execution of contractual agreements, reducing the need for intermediaries.

* **Supply Chain Management**

Blockchain enhances transparency and traceability in supply chains by recording and verifying every transaction in the production and distribution process. This helps in reducing fraud, ensuring authenticity, and improving efficiency.

* **Cross-Border Payments**

Blockchain facilitates faster and more cost-effective cross-border payments by eliminating intermediaries and reducing transaction processing times.

* **Digital Identity**

Blockchain can be used for secure and decentralized identity management. It provides individuals with control over their personal information, reducing the risk of identity theft.

* **Tokenization of Assets**

Blockchain enables the tokenization of real-world assets such as real estate, art, or commodities. This allows for fractional ownership and increased liquidity.

**Benefits of Using Web3 for Ethereum Development**

Web3.py is a Python library that offers several benefits for Ethereum development, making it a popular choice for developers looking to create decentralized applications and interact with the Ethereum blockchain.

Some of the benefits of using Web3.py for Ethereum development include:

* **Convenience:** Web3.py provides a convenient and easy-to-use interface for interacting with smart contracts, managing Ethereum accounts, sending transactions, and querying blockchain data.
* **Flexibility:** Python is a trusted programming language used by thousands of developer’s worldwide, offering flexibility for various use cases.
* **Compatibility**: Web3.py allows developers to work with Ethereum using Python, making it accessible to a wide range of developers familiar with python
* **Connection to the Ethereum network:** Web3.py allows developers to connect to the Ethereum network using a connection provider like Infura.

**Inter-Planetary File System**

The Inter-Planetary File System (IPFS) is a peer-to-peer (P2P) distributed system for storing, accessing, and sharing files, websites, applications, and data. IPFS is built around the principles of content addressing and peer-to-peer networking, and it is designed to solve some of the problems with the internet and current protocols like Hypertext transfer protocol.

* **Content addressing:** Every file added to IPFS is given a unique address derived from a hash of the file's content. This address is called a Content Identifier (CID), and it combines the hash of the file and a unique identifier. CIDs are unique to the data from which they were computed, which provides IPFS with several benefits, such as the ability to fetch data based on its content rather than its location.
* **Peer-to-peer networking:** IPFS is a distributed file storage protocol that allows computers all over the globe to store and serve files as part of a giant peer-to-peer network. Any computer, anywhere in the world, can download the IPFS software and start hosting and serving files. If someone runs IPFS on their computer and uploads a file to the IPFS network, that file can be viewed and downloaded by anyone else in the world who is also running IPFS.
* **CID and IPF**S: IPFS uses content-addressing to uniquely identify each file in a global namespace connecting IPFS. IPFS can replace the location-based hypermedia server protocols HTTP and HTTPS. IPFS allows users to host and receive content in a manner similar to BitTornt, and it aims to create a network.

**How does IPFS ensure data integrity and authenticity?**

IPFS ensures data integrity and authenticity through the use of cryptographic hash functions and public-key cryptography. When a file is added to the IPFS network, it is divided into smaller chunks, and these chunks are then used to create a MerkleDAG (Directed Acyclic Graph), which represents the file’s structure.

IPFS ensures data authenticity through the use of public-key cryptography. Each file in IPFS is associated with a unique public key, which is used to sign the file’s metadata. This signature provides a proof that the file was created by the owner of the corresponding private key. When a user retrieves a file from IPFS, they can verify the authenticity of the file by checking the signature against the associated public key. If the signature is valid, it means that the file has not been tampered with and can be trusted.

**5.2 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between inputs and outputs. Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionalities that define what a system is supposed to accomplish. These are the requirements that the user specifically demands as basic facilities that the system should offer.

**5.3.1 User Registration and Management**

* Suppliers and customers should be able to create new user accounts with basic details like name, email, password etc.
* They should be able to update their profiles or delete their accounts.
* The system needs to securely store user credentials and data.
* Different access privileges for suppliers’ vs customers.

**5.3.2 Product Catalog Management**

* Suppliers can add new products to sell, providing product images, description, pricing and inventory counts.
* They can edit or update details for existing product listings.
* Appropriate validations during product addition (pricing, inventory > 0 etc).
* Catalog organized by categories for easy browsing.

**5.3.3 Shopping Cart and Order Management**

* Customers can search products, view details, add them to a cart
* They can specify product quantity within available inventory
* Checkout to place the order by providing address, payment details
* View status and history of orders placed

**5.3.4 Inventory Status and Availability Checks**

* System checks product inventory counts when orders are placed
* Won't allow orders if item inventory insufficient
* Dashboard for suppliers to see inventory status across products
* The system employs real-time updates to ensure accurate and up-to-date.

**5.3.5 Payment Processing**

* Integrate payment gateways like Stripe to allow card payments
* Process payments during order checkout, handle failures
* Orders marked Paid once successful, unlocking fulfillment

**5.3 NON-FUNCTIONAL REQUIREMENTS:**

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs.

Example, the site should load in 3 seconds when the no of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

1. **Security**

Employ advanced encryption techniques to safeguard sensitive information against unauthorized access. Leverage the inherent security features of Blockchain, including immutability and decentralized storage. Immutability ensures that once a transaction is recorded on the blockchain, it cannot be modified or deleted, ensuring the integrity of the data and preventing unauthorized access.

1. **Performance**
   * + **Scalability:** Design the system to scale gracefully, accommodating a growing user base and transaction volume.
     + **Optimized** Response Time: Ensure low-latency interactions to enhance user experience and system responsiveness.
2. **Usability**
   * + **Intuitive Design:** Focus on crafting an intuitive user interface that prioritizes ease of navigation and interaction.
     + **User Training Materials**: Develop comprehensive training materials to aid users in familiarizing themselves with the system.
3. **Reliability**
   * + **High Availability**: Strive for a high level of system availability by capitalizing on the decentralized nature of Blockchain.
     + **Effective Error Handling:** Implement robust error-handling mechanisms to promptly address and rectify issues, enhancing overall system reliability
     + **Fault tolerance:** Specify the system's ability to continue functioning in the presence of faults or errors.

**CHAPTER 5**

**REQUIREMENT SPECIFICATIONS**

**3.1 HARDWARE REQUIREMENTS**

* + - SYSTEM : Intel Core i3
    - HARD Disk : 120 GB
    - MONITOR : 15”LED
    - INPUT DEVICES : Keyboard, Mouse
    - RAM : 4 GB

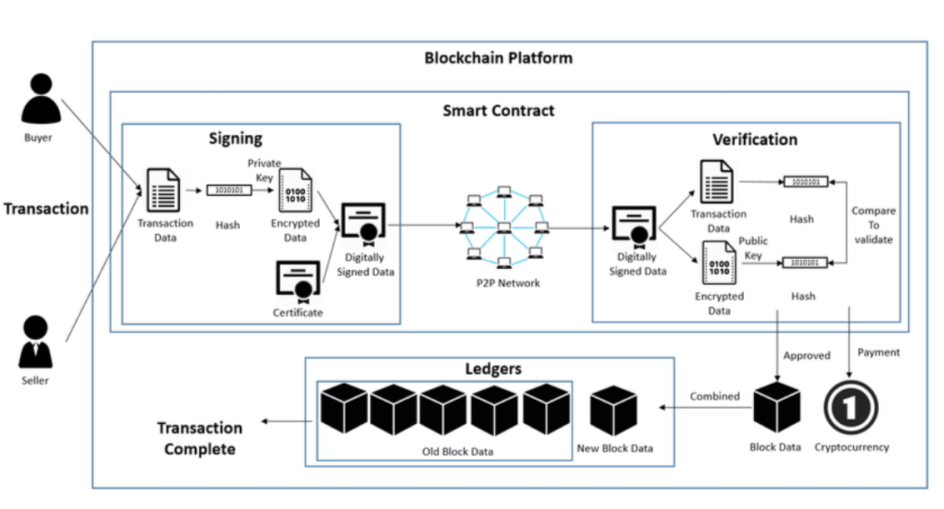
**3.2 SOFTWARE REQUIREMENTS**

* + - OPERATING SYSTEM : Windows 7 or Above
    - CODING LANGUAGE : Python
    - DEBUGGER/EMULATOR : Any Browser

**CHAPTER 6**

**SOFTWARE DESIGN**

**6.1 ARCHITECTURAL DESIGN**

****

**Figure: - 6.1 System Architecture**

**6.2 MODULES**

**Login Module**

* **User Authentication:** Implement a robust authentication mechanism to verify the identity of both product suppliers and customers.
* **Authorization:** Introduce role-based access controls to restrict and manage user access privileges effectively

**Signup Module**

* **User Registration:** Develop a streamlined registration process, allowing customers and suppliers to onboard by providing necessary details.
* **Credential Management:** Assign unique usernames and passwords securely for authenticated access.

**Add Product Module**

* **Blockchain Integration:** Incorporate Ethereum with Truffle to harness the power of Blockchain for secure storage and management of product details.
* **Image Storage:** Leverage IPFS to store product images, ensuring data integrity and accessibility through unique hash codes.

**Update Quantity Module**

* **Supplier Access:** Facilitate suppliers in updating product quantities securely on the Blockchain.
* **Blockchain Verification:** Implement a robust verification mechanism to ensure the integrity of data during quantity updates.

**View Orders Module**

* **Order Tracking:** Empower suppliers with the ability to view and track customer orders, promoting efficient order management.
* **Blockchain Integration:** Store order data on the Blockchain for enhanced security and immutability.

**Browse Products Module**

* **User Interface Design:** Craft an intuitive and user-friendly interface, allowing customers to seamlessly browse, search, and interact with products.
* **Order Placement:** Enable customers to place orders efficiently through a user-centric interface.

**Integration of Payment Gateways**

By seamlessly integrating reputable payment gateways like Stripe, the system empowers users with a secure and reliable card payment infrastructure. This strategic inclusion ensures a standardized and trustworthy financial transaction experience. Leveraging the robust features of these gateways, the system upholds industry standards, reinforcing user confidence in the security of their payments.

**Immutable Ledgers**

At the core of blockchain's impact is its utilization of distributed ledgers, establishing an incorruptible and transparent record of all transactions. This not only enhances security but also ensures accountability and trust in the e-commerce ecosystem. In essence, immutable ledgers lay the foundation for a future of trustless commerce. By eliminating the need for centralized intermediaries and relying on a secure, shared record-keeping system, blockchain technology paves the way for a more transparent and equitable e-commerce landscape, benefiting both consumers and businesses alike.

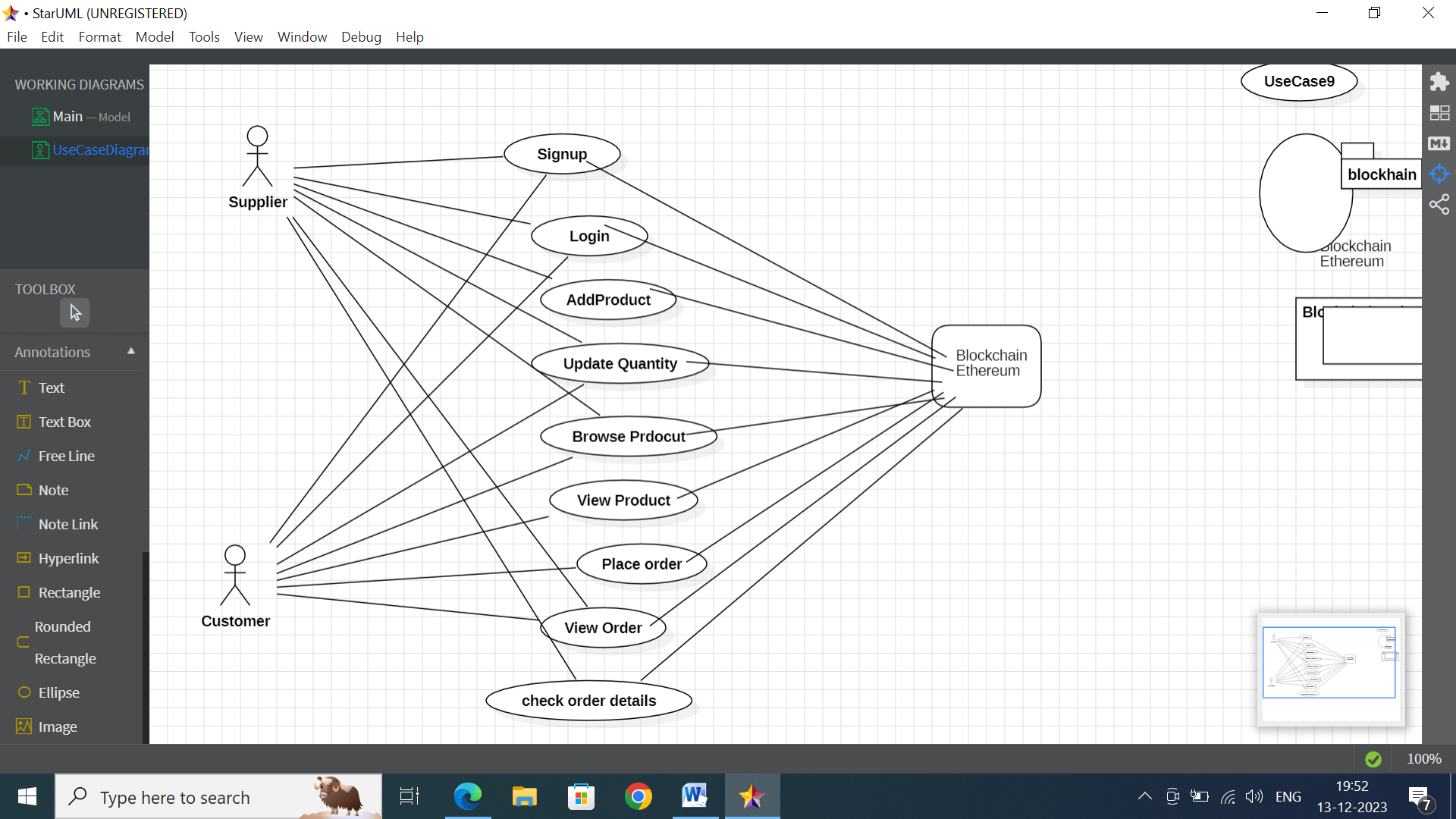
**6.3 UML DIAGRAMS**

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing, and documenting the artifacts of a software system, 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 modeling of large and complex systems.

**6.3.1. Use Case Diagram**

A use-case diagram in the Unified Modeling Language (UML) is a pivotal component derived from thorough use-case analysis, serving as a visual representation to offer a comprehensive overview of a system's functionality. This diagram encapsulates the interactions between various actors and the specific goals they aim to achieve, represented as individual use cases. Beyond merely showcasing the functionalities, it delves into illustrating any interdependencies that may exist between these use cases.

The primary objective of a use-case diagram extends beyond a mere enumeration of system functions; it serves as a powerful tool for depicting the dynamic relationships and collaborations among different actors and use cases.

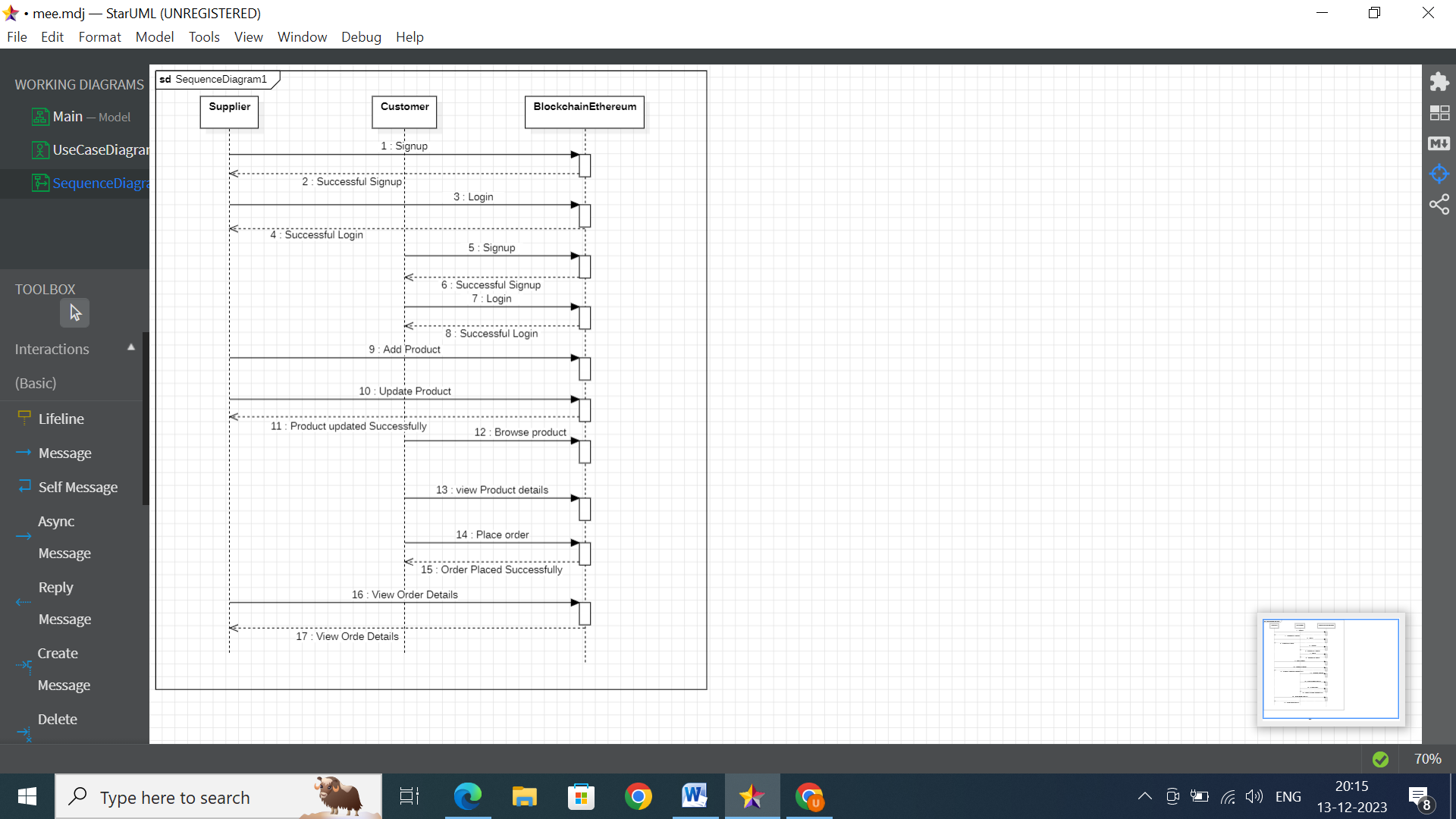
****

**Figure: - 6.2 Use-Case Diagram**

The use case diagram outlines the step-by-step interactions within the blockchain-based system. Users initiate by signing up, creating secure identities on the blockchain. Following login, customers explore products, view detailed information, and place orders with specified quantities and options. The system accommodates order history viewing for customers. On the supplier side, the addition of new products, updating quantities, and viewing orders are seamlessly integrated. The entire process is underpinned by blockchain technology, guaranteeing secure data storage and tracking, thereby upholding the integrity of all transactions within the system. This comprehensive use case flow ensures a robust and transparent experience for both customers and suppliers in the blockchain ecosystem.

**6.3.2 Sequence Diagram**

A sequence diagram in Unified Modeling 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

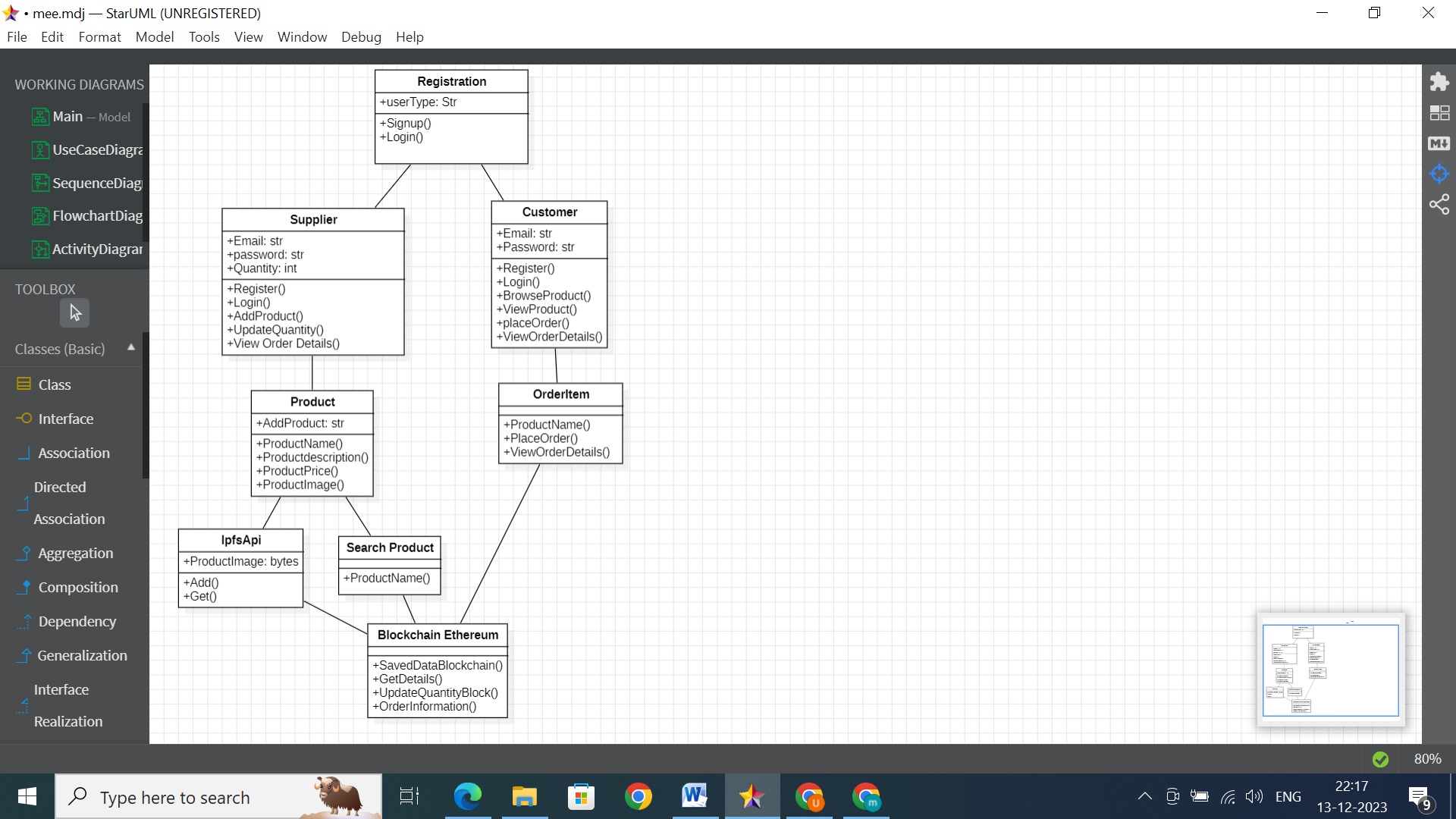
****

**Figure: - 6.3 Sequence Diagram**

The provided sequence diagram outlines the process of adding a product to an e-commerce system using blockchain technology. Initiated by the supplier, a "Add Product" message is sent to the blockchain, encompassing essential product details. The blockchain responds with a confirmation of successful product creation, and the system logs this event for auditing purposes. Additional elements in the diagram, such as "Successful Signup" and "Login," seem to pertain to separate actions, potentially related to user authentication. The interactions "Update Product" and its corresponding confirmation suggest the ability to modify existing product information, while "View Product Details" allows the supplier to inspect the newly created or updated product. Notably, interactions like "Place Order" and "View Order Details" appear unrelated to the primary focus of product addition, introducing complexity and potential confusion to the diagram.

**6.3.3 Class Diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) plays a fundamental role as a static structure diagram, providing a detailed representation of a system's architecture. This visualization goes beyond a mere enumeration of classes; it comprehensively outlines the attributes and operations

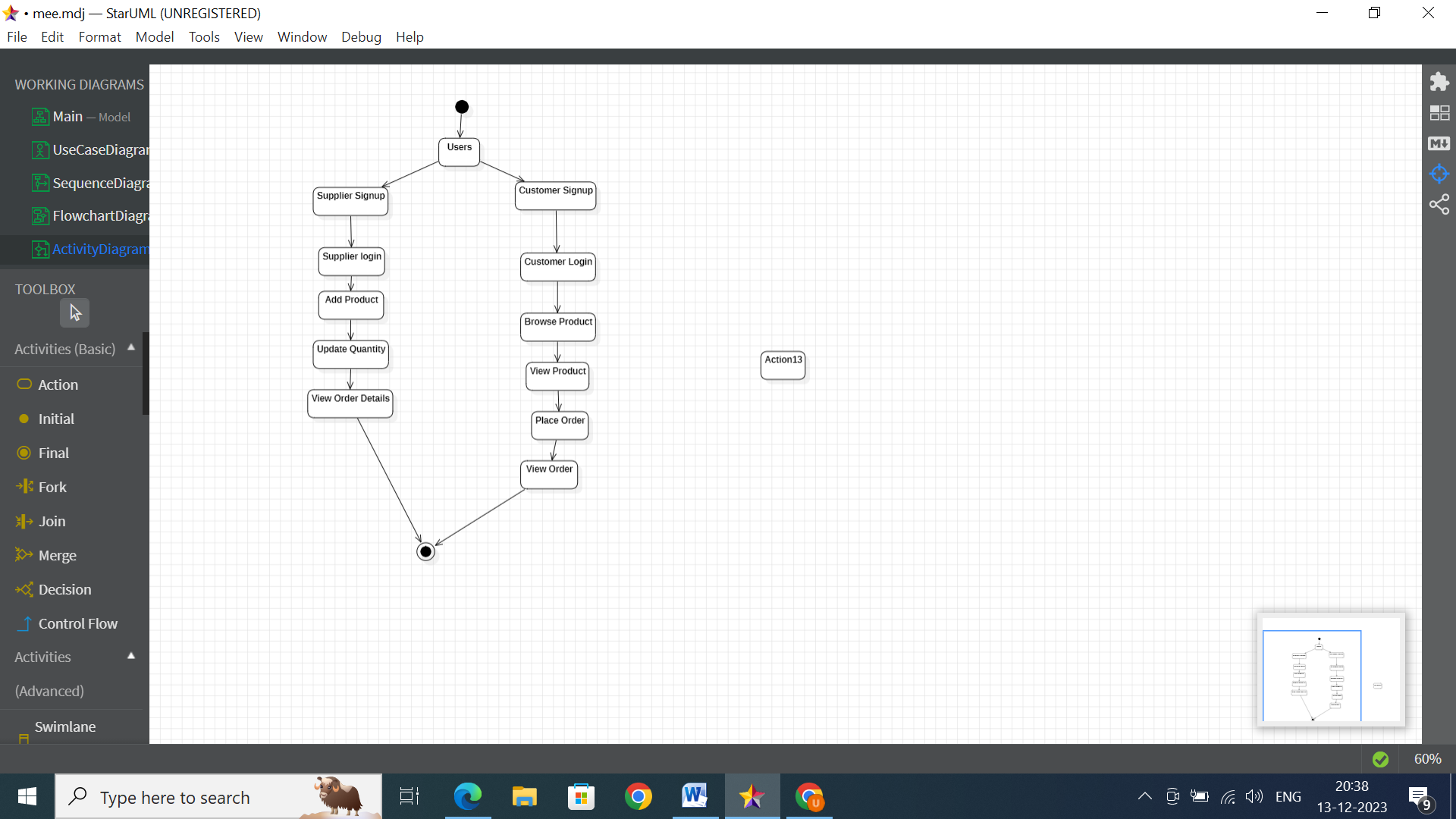
****

**Figure: - 6.4 Class Diagram**

The Ecommerce Application orchestrates the entire system, utilizing Web3Helper for blockchain interactions, IPFS for image uploading and management. Registration Stores user information, while Product represents product details with an associated IPFS hash for images. Orders are managed by the Order class, which contains Order Items linking specific products and quantities handles file read/write operations. Web3 interacts with the blockchain for authentication and data updates.IPFS interactions, allowing the storage and retrieval of product images. Supplier can AddProdcuts ,Update them And Get The Customer Order details, where Customer Can Search Products and Place the Order and All the details are stored in Blockchain.

**6.3.4 Activity Diagram**

An activity diagram, classified as a behavior diagram in the Unified Modeling Language (UML), functions as a dynamic flowchart that visually maps the progression from one activity to another within a system or process. Serving as an invaluable tool for depicting dynamic aspects, it outlines the sequential steps and interactions essential for understanding the operational aspects of a modeled system.



**Figure:-6.5 Activity Diagram**

The activity diagram outlines the user-friendly process of adding a product to an e-commerce system, involving the user (supplie) and the system. Starting with the user clicking "Add Product," the steps systematically unfold, including inputting product information, optional features like image uploads, and validation checks. The system then saves validated data to the database, signaling completion with a confirmation message. Users can view the product list or add more products, with optional paths for canceling or addressing image size issues.Distinguish user and system actions, and decision diamonds manage conditional flows, emphasizing the systematic and intuitive nature of the product addition workflow.

**CHAPTER 7**

**SOURCE CODE**

**VIEW.PY**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

import pymysql

from django.http import HttpResponse

from django.core.files.storage import FileSystemStorage

import os

from datetime import date

import os

import json

from web3 import Web3, HTTPProvider

import ipfsApi

import os

from django.conf import settings

from django.core.files.storage import FileSystemStorage

import pickle

global details, username

details=''

global contract

api = ipfsApi.Client(host='http://127.0.0.1', port=5001)

def readDetails(contract\_type):

global details

details = ""

print(contract\_type+"======================")

blockchain\_address = 'http://127.0.0.1:9545' #Blokchain connection IP

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'Ecommerce.json' #ecommerce contract code

deployed\_contract\_address '0x8AFB715a81922908F1D3F59a89f24F4601B771DC' #hash address to access student contract

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi) #now calling contract to access data

if contract\_type == 'signup':

details = contract.functions.getUser().call()

if contract\_type == 'addproduct':

details = contract.functions.getProduct().call()

if contract\_type == 'bookorder':

details = contract.functions.getOrder().call()

print(details)

def saveDataBlockChain(currentData, contract\_type):

global details

global contract

details = ""

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'Ecommerce.json' #ecommerce contract file

deployed\_contract\_address ='0x8AFB715a81922908F1D3F59a89f24F4601B771DC' #contract address

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

readDetails(contract\_type)

if contract\_type == 'signup':

details+=currentData

msg = contract.functions.addUser(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract\_type == 'addproduct':

details+=currentData

msg = contract.functions.addProduct(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract\_type == 'bookorder':

details+=currentData

msg = contract.functions.bookOrder(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def updateQuantityBlock(currentData):

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'Ecommerce.json' #student contract file

deployed\_contract\_address'0x8AFB715a81922908F1D3F59a89f24F4601B771DC' #contract address

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract =web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

msg = contract.functions.addProduct(currentData).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def BrowseProducts(request):

if request.method == 'GET':

output='<tr><td><fontsize=""> color="black">Product&nbsp;Name</font></td><td><select name="t1">'

readDetails("addproduct")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == 'addproduct':

output+='<option value="'+arr[2]+'">'+arr[2]+'</option>'

output+="</select></td></tr>"

context= {'data1':output}

return render(request, 'BrowseProducts.html', context)

def Login(request):

if request.method == 'GET':

return render(request, 'Login.html', {})

def ViewOrders(request):

if request.method == 'GET':

global details

user = ''

with open("session.txt", "r") as file:

for line in file:

user = line.strip('\n')

file.close()

output = '<table border=1 align=center>'

output+='<tr><th><font size=3 color=black>Product Name</font></th>'

output+='<th><font size=3 color=black>Customer Name</font></th>'

output+='<th><font size=3 color=black>Contact No</font></th>'

output+='<th><font size=3 color=black>Email ID</font></th>'

output+='<th><font size=3 color=black>Address</font></th>'

output+='<th><font size=3 color=black>Ordered Date</font></th></tr>'

readDetails("bookorder")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == 'bookorder':

print(arr[2]+" "+user)

details = arr[3].split(",")

pid = arr[1]

user = arr[2]

book\_date = arr[4]

output+='<tr><td><font size=3 color=black>'+pid+'</font></td>'

output+='<td><font size=3 color=black>'+user+'</font></td>'

output+='<td><font size=3 color=black>'+details[0]+'</font></td>'

output+='<td><font size=3 color=black>'+details[1]+'</font></td>'

output+='<td><font size=3 color=black>'+details[2]+'</font></td>'

output+='<td><font size=3color=black>'+str(book\_date)+'</font></td></tr>'

output+="</table><br/><br/><br/><br/><br/><br/>"

context= {'data':output}

return render(request, 'ViewOrders.html', context)

def Register(request):

if request.method == 'GET':

return render(request, 'Register.html', {})

def AddProduct(request):

if request.method == 'GET':

return render(request, 'AddProduct.html', {})

def BookOrder(request):

if request.method == 'GET':

global details

pid = request.GET['crop']

user = ''

with open("session.txt", "r") as file:

for line in file:

user = line.strip('\n')

file.close()

readDetails("signup")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

if arr[1] == user:

details = arr[3]+","+arr[4]+","+arr[5]

break

today = date.today()

data = "bookorder#"+pid+"#"+user+"#"+details+"#"+str(today)+"\n"

saveDataBlockChain(data,"bookorder")

output = 'Your Order details Updated<br/>'

context= {'data':output}

return render(request, 'ConsumerScreen.html', context)

def UpdateQuantity(request):

if request.method == 'GET':

output = ''

user = ''

with open("session.txt", "r") as file:

for line in file:

user = line.strip('\n')

file.close()

output = '<tr><td><font size="" color="black">Product&nbsp;Name</font></td><td><select name="t1">'

readDetails("addproduct")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "addproduct":

if arr[1] == user:

output+='<option value="'+arr[2]+'">'+arr[2]+'</option>'

output+="</select></td></tr>"

context= {'data':output}

return render(request, 'UpdateQuantity.html', context)

def SearchProductAction(request):

if request.method == 'POST':

ptype = request.POST.get('t1', False)

output = '<table border=1 align=center>'

output+='<tr><th><font size=3 color=black>Supplier Name</font></th>'

output+='<th><font size=3 color=black>Product Name</font></th>'

output+='<th><font size=3 color=black>Price</font></th>'

output+='<th><font size=3 color=black>Quantity</font></th>'

output+='<th><font size=3 color=black>Description</font></th>'

output+='<th><font size=3 color=black>Image</font></th>'

output+='<th><font size=3 color=black>Purchase Product</font></th></tr>'

readDetails("addproduct")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

print("my=== "+str(arr[0])+" "+arr[1]+" "+arr[2]+" "+ptype)

if arr[0] == 'addproduct':

if arr[2] == ptype:

output+='<tr><td><font size=3 color=black>'+arr[1]+'</font></td>'

output+='<td><font size=3 color=black>'+arr[2]+'</font></td>'

output+='<td><font size=3 color=black>'+str(arr[3])+'</font></td>'

output+='<td><font size=3 color=black>'+str(arr[4])+'</font></td>'

output+='<td><font size=3 color=black>'+arr[5]+'</font></td>'

content = api.get\_pyobj(arr[6])

content = pickle.loads(content)

if os.path.exists('C:/Users/DELL/Desktop/pro/13.Blockchain Based Ecommerce online application/13.Blockchain Based Ecommerce online application/BlockchainEcommerce/BlockchainEcommerce/EcommerceApp/static/product.png')

output+='<td><img src="/static/product.png"

width="200" height="200"></img></td>'

output+='<td><a href=\'BookOrder?farmer='+arr[1]+'&crop='+arr[2]+'\'><font

size=3 color=black>Click Here</font></a></td></tr>'

output+="</table><br/><br/><br/><br/><br/><br/>"

context= {'data':output}

return render(request, 'SearchProducts.html', context)

def QuantityUpdateAction(request):

if request.method == 'POST':

pname = request.POST.get('t1', False)

qty = request.POST.get('t2', False)

user = '' ''

with open("session.txt", "r") as file:

for line in file:

user = line.strip('\n')

file.close()

index = 0

record = ''

readDetails("addproduct")

rows = details.split("\n")

tot\_qty = 0

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "addproduct":

if arr[1] == user and arr[2] == pname:

tot\_qty = int(arr[4])

tot\_qty = tot\_qty + int(qty)

index = i

record = arr[0]+"#"+arr[1]+"#"+arr[2]+"#"+arr[3]+"#"+str(tot\_qty)+"#"+arr[5]+"#"+arr[6]+"\n"

break

for i in range(len(rows)-1):

if i != index:

record += rows[i]+"\n"

updateQuantityBlock(record)

context= {'data':"Quantity details updated & new available quantity: "+str(tot\_qty)}

return render(request, 'SupplierScreen.html', context)

def AddProductAction(request):

if request.method == 'POST':

cname = request.POST.get('t1', False)

qty = request.POST.get('t2', False)

price = request.POST.get('t3', False)

desc = request.POST.get('t4', False)

if 't5' in request.FILES:

# 't5' file upload present

image = request.FILES['t5'] # Get the uploaded file

hashcode = api.add(image)

else:

# No 't5' file upload

image = None

imagename = None

user = '' “

with open("session.txt", "r") as file:

user = file.read().strip()

if image:

# Image was uploaded

myfile = pickle.dumps(image)

hashcode = api.add\_pyobj(myfile)

data = f"addproduct#{user}#{cname}#{price}#{qty}#{desc}#{hashcode}\n"

saveDataBlockChain(data,"addproduct")

context = {

'data': f"Product details saved and IPFS image storage hashcode = {hashcode}"

}

else:

# No image uploaded

data = f"addproduct#{user}#{cname}#{price}#{qty}#{desc}#NoImageUploaded\n"

saveDataBlockChain(data,"addproduct")

context = {

'data': "Product details saved without image"

}

return render(request, 'AddProduct.html', context)

def Signup(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

contact = request.POST.get('contact', False)

email = request.POST.get('email', False)

address = request.POST.get('address', False)

usertype = request.POST.get('type', False)

record = 'none'

readDetails("signup")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

if arr[1] == username:

record = "exists"

break

if record == 'none':

data = "signup#"+username+"#"+password+"#"+contact+"#"+email+"#"+address+"#"+usertype+"\n"

saveDataBlockChain(data,"signup")

context= {'data':'Signup process completd and record saved in Blockchain'}

return render(request, 'Register.html', context)

else:

context= {'data':username+'Username already exists'}

return render(request, 'Register.html', context)

def UserLogin(request):

if request.method == 'POST':

username = request.POST.get('username', False)

password = request.POST.get('password', False)

usertype = request.POST.get('type', False)

status = 'none'

readDetails("signup")

rows = details.split("\n")

for i in range(len(rows)-1):

arr = rows[i].split("#")

if arr[0] == "signup":

if arr[1] == username and arr[2] == password and arr[6] == usertype:

status = 'success'

break

if status == 'success' and usertype == 'Supplier':

file = open('session.txt','w')

file.write(username)

file.close()

context= {'data':"Welcome "+username}

return render(request, 'SupplierScreen.html', context)

elif status == 'success' and usertype == 'Consumer':

file = open('session.txt','w')

file.write(username)

file.close()

context= {'data':"Welcome "+username}

return render(request, 'ConsumerScreen.html', context)

else:

context= {'data':'Invalid login details'}

return render(request, 'Login.html', context)

**CHAPTER 8**

**SYSTEM TESTING**

* 1. **TEST CASES:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **1.** | Signup | Supplier Signup with valid details | Supplier account is successfully created | Supplier Account Created Successfully | Pass |

**Table No.8.1 Test case 1 Supplier Signup**

This test case validates the signup functionality for suppliers by entering valid details. The expected outcome is the successful creation of a supplier account. The test was executed, and the actual result confirms that the supplier account was created successfully, leading to a Pass status for the test case.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **2.** | Signup | Customer Signup with valid details | Customer account is successfully created | Customer Account Created Successfully | Pass |

**Table No.8.2 Test case 2 Customer Signup**

This test case validates the signup functionality for suppliers by entering valid details. The expected outcome is the successful creation of a supplier account. The test was executed, and the actual result confirms that the supplier account was created successfully, leading to a Pass status for the test case.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **3.** | Login | Supplier Login with valid details | Supplier is successfully Logged in | Supplier Logged in Successfully | Pass |

**Table No.8.3 Test case 3 Supplier Login**

This test case evaluates the login functionality for suppliers using valid login credentials. The expected outcome is the successful login of the supplier. The test was executed, and the actual result confirms that the supplier logged in successfully, leading to a Pass status for the test case.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **4.** | Login | Customer Login with valid details | Supplier is successfully Logged in | Supplier Logged in Successfully | pass |

**Table No.8.4 Test case 4 Customer Login**

This test case evaluates the login functionality for suppliers using valid login credentials. The expected outcome is the successful login of the supplier. The test was executed, and the actual result confirms that the supplier logged in successfully, leading to a Pass status.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **5.** | Login | Attempts to Login with Invalid details | Login Fails with Error Message | Error Message Appear “invalid username or password | Pass |

**Table No.8.5 Test case 5 Invalid login**

This test case examines the login functionality when attempting to log in with invalid details. The expected outcome is a failed login with an appropriate error message. The test was executed, and the actual result confirms that the error message "Invalid username or password" appeared, meeting the expected result and resulting in a Pass status for the test case.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **6.** | Add  Product | Supplier adds a new product details | Product successfully added and recorded on blockchain | Product added Successfully | pass |

**Table No.8.6 Test case 6 Add Product**

The expected outcome is the successful addition of the product to the marketplace, with the transaction securely recorded on the blockchain. The test verifies the system's capability to add products effectively, and success is confirmed when the system appropriately acknowledges the completion of the product addition process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **7.** | Update Quantity | Supplier updates Quantity for an Existing product | Product Quantity is successfully updated on Blockchain | Product Quantity Updated Successfully | pass |

**Table No.8.7 Test case 7 Update Quantity**

In this test case, the objective is to confirm the successful update of a product quantity on the blockchain when a supplier initiates the process. The scenario involves a supplier with proper authorization updating the quantity of an existing product. The expected result is the accurate recording of the updated quantity information on the blockchain, and the test passes if these steps are completed without errors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **8.** | View Orders | Supplier Views list of orders | Supplier is able to see all orders placed by Customers | Supplier Able to See list of orders | pass |

**Table No.8.8 Test case 8 View Orders**

. The scenario involves the supplier accessing the system to view a comprehensive list of orders placed by customers. The expected result is that the supplier can successfully see all relevant details of the orders, and the pass criteria indicate that the test is considered successful if the supplier encounters no issues and can readily access the complete list of customer orders.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **9.** | Browse Product | Customer Search For an Existing product | Relevant Product details are displayed to the customer | Correct Product details displayed | pass |

**Table No.8.9 Test case 9 Browse Product**

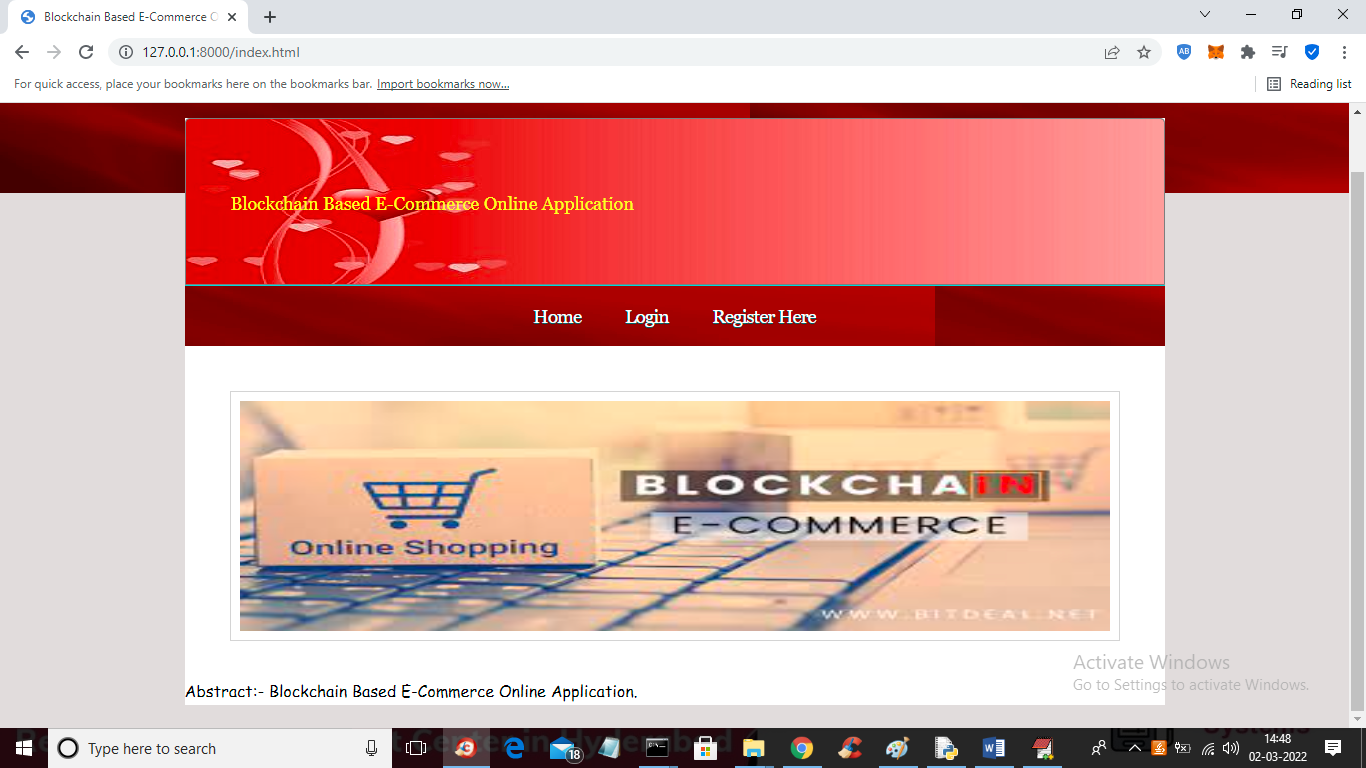
In this test case, the customer initiates a product search through the "Browse Product" feature. The system is expected to display accurate details of the relevant product. The test is considered successful (pass) if the displayed information aligns correctly with the searched product, confirming the effectiveness of the product browsing functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Description** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| **10.** | Browse Product | Customer Search For an non-Existing product | No result are found | Message “No results found” | pass |

**Table No.8.10 Test case 10 Browse Product**

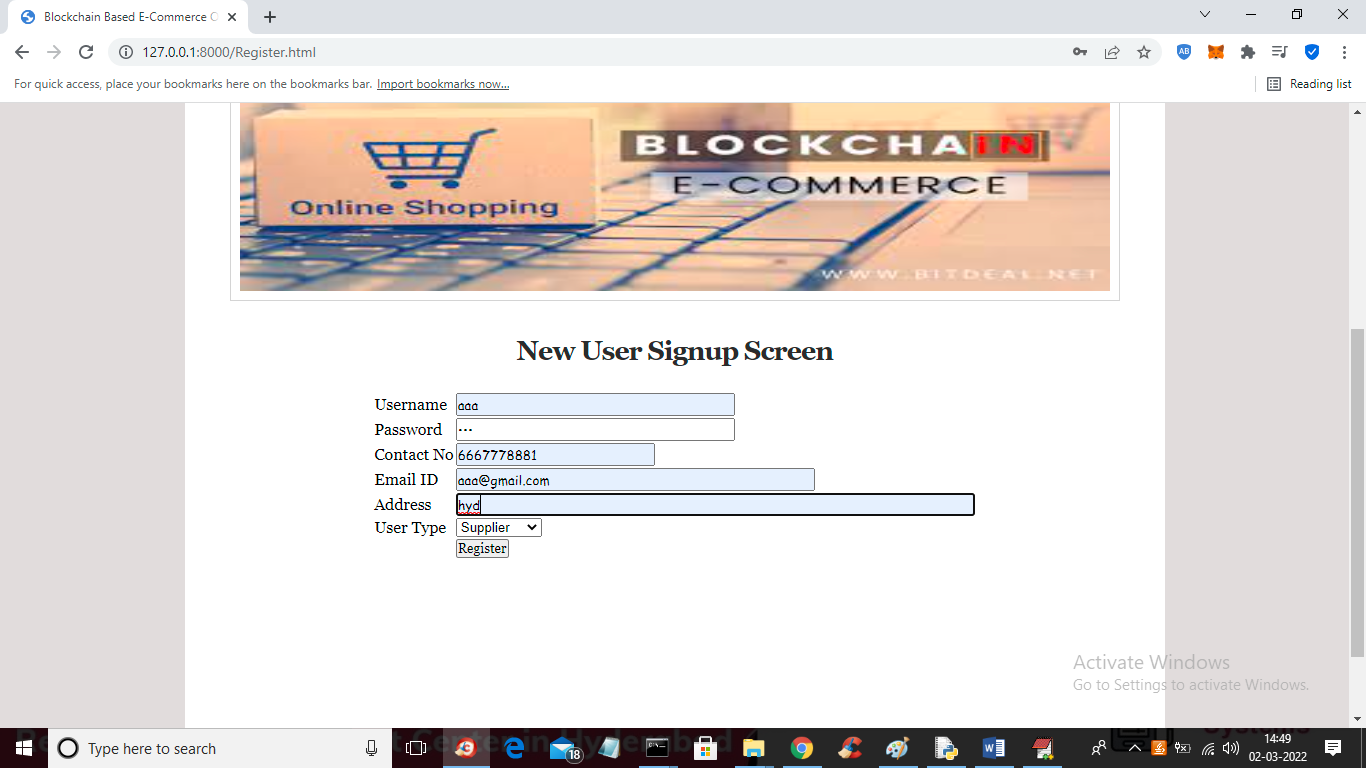
This test case involves a customer utilizing the "Browse Product" feature to search for a non-existing product. The expected outcome is that the system appropriately handles the absence of search results and displays a message stating "No results found." The test is considered successful (pass) if the system accurately communicates the lack of matching products, providing a clear indication to the customer.

**CHAPTER 9  
OUTPUT SCREENS**



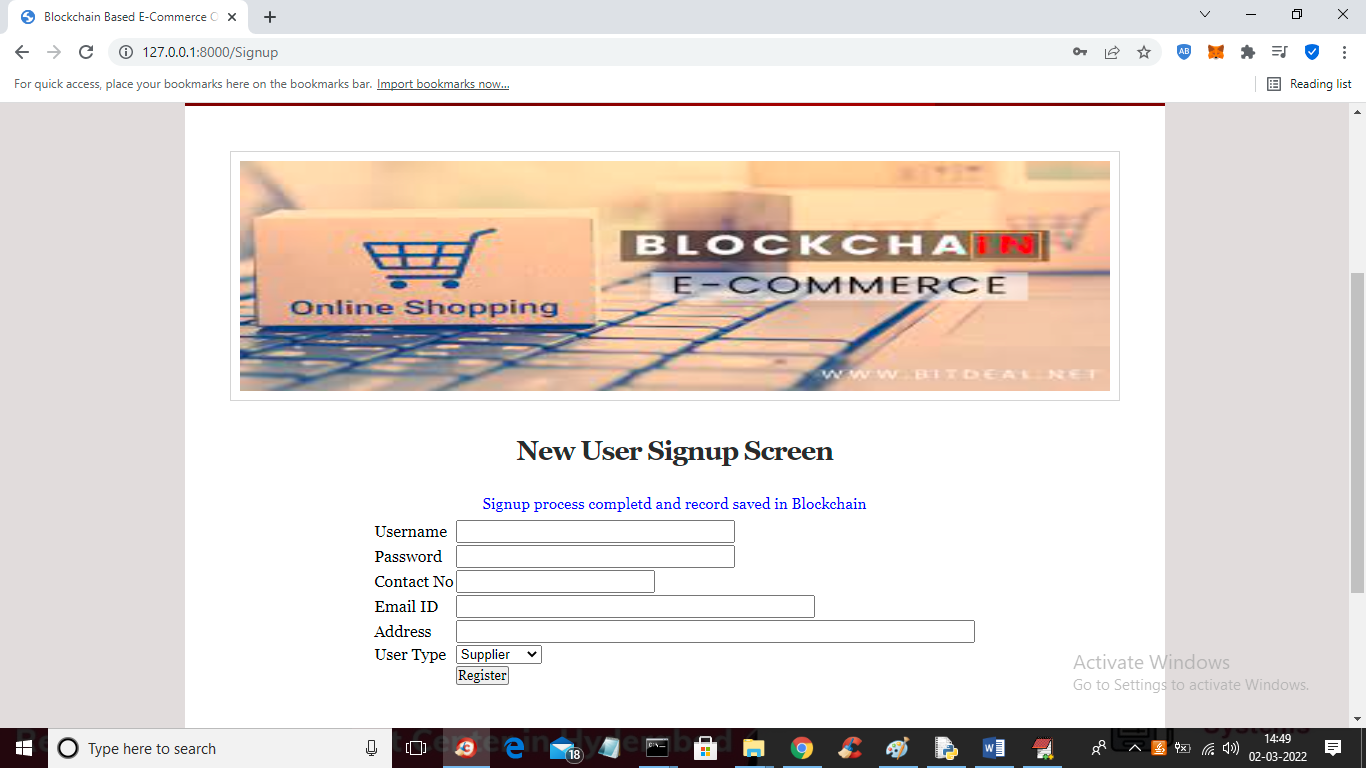
**Figure: - 9.1 User Signup**

In above screen click on ‘Register here’ link to sign up two users such as consumer and supplier



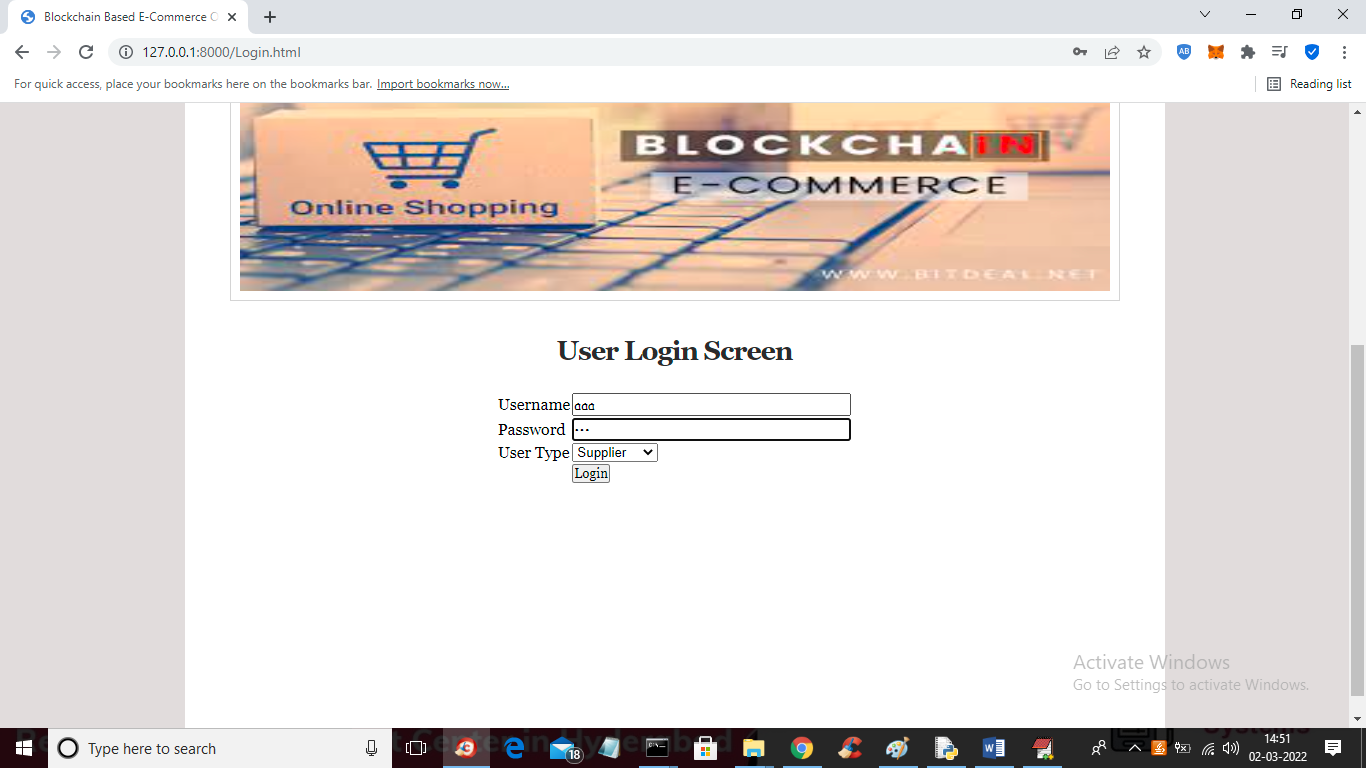
**Figure:-9.2 Suppler Signup**

In above screen supplier is getting signup and then press Register button to get below screen



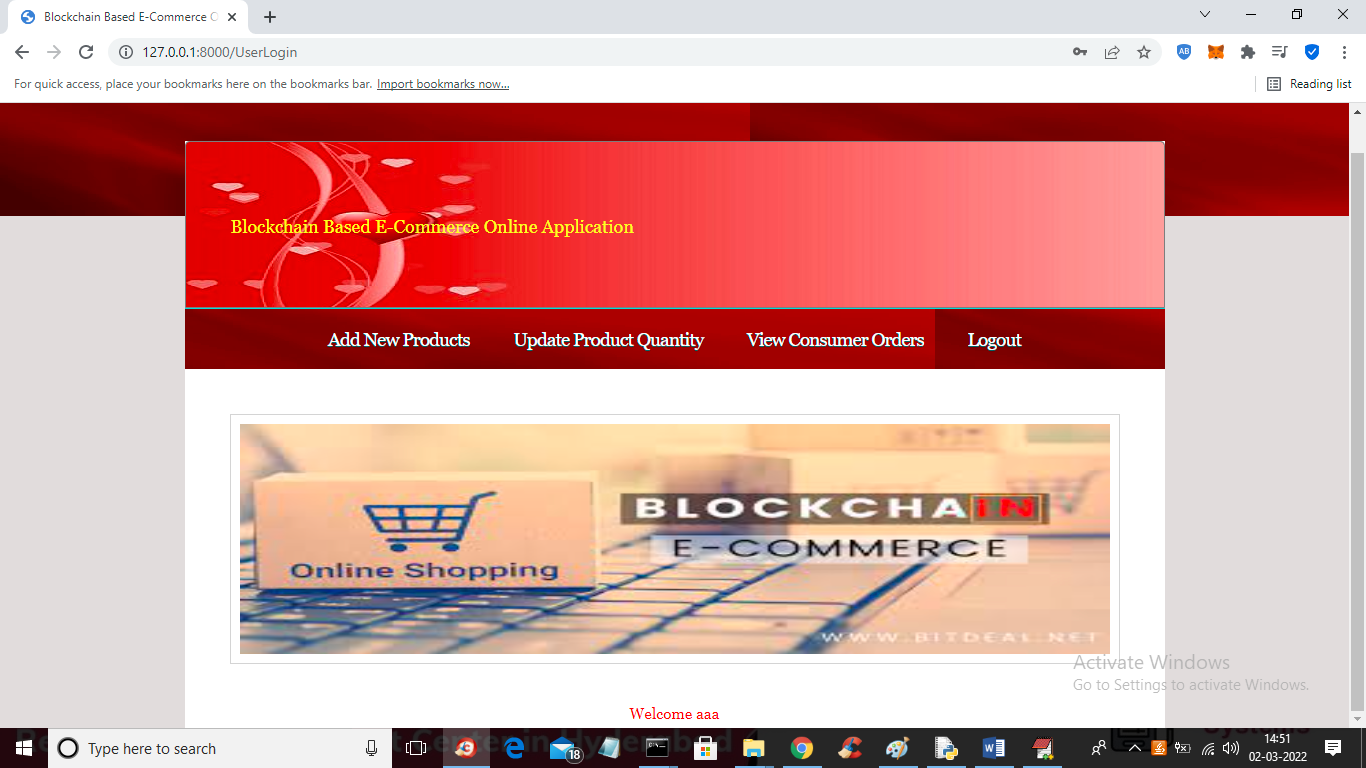
**Figure:-9.3 Supplier Signup Completed**

In above screen signup completed and now add consumer user



**Figure:-9.4 Supplier Login**

In above screen supplier is login and after login will get below screen



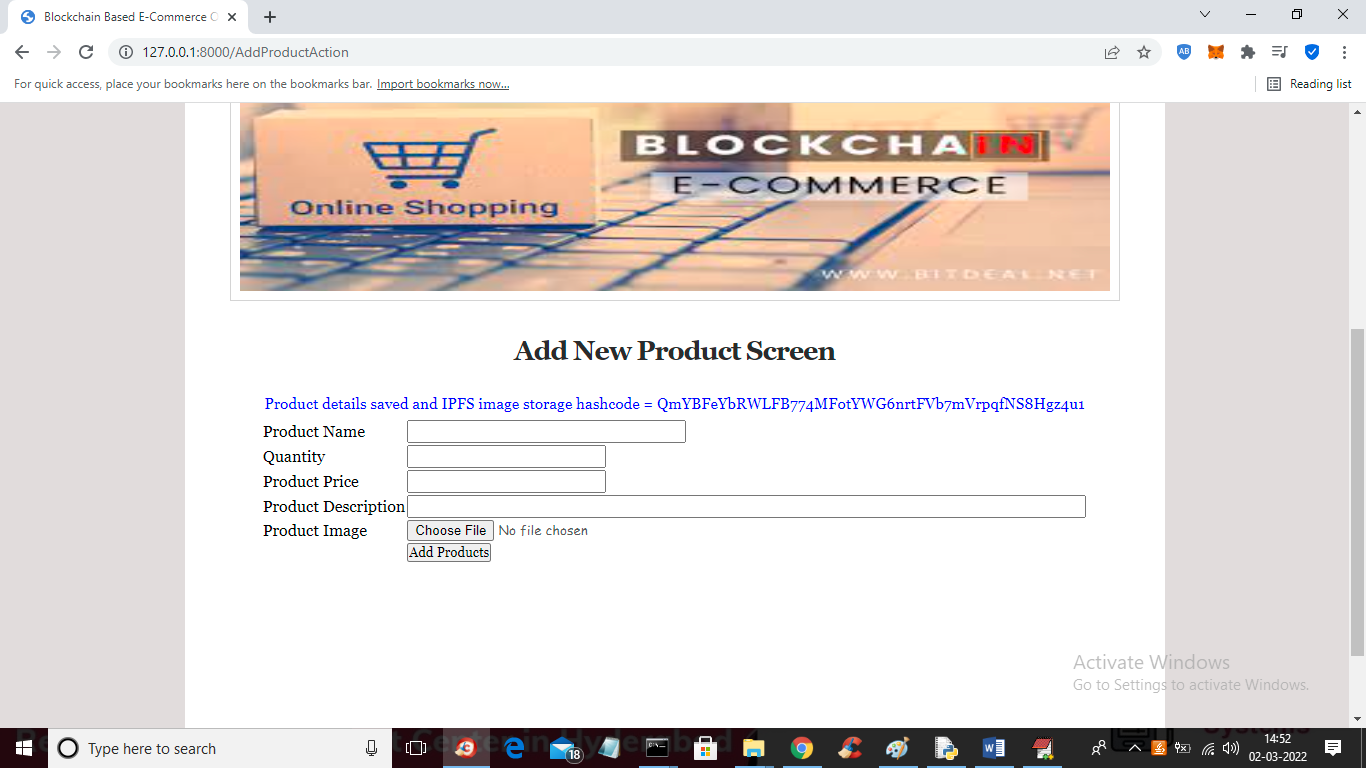
**Figure:-9.5 Add products**

in above screen click on ‘Add New Products’ link to add new product details



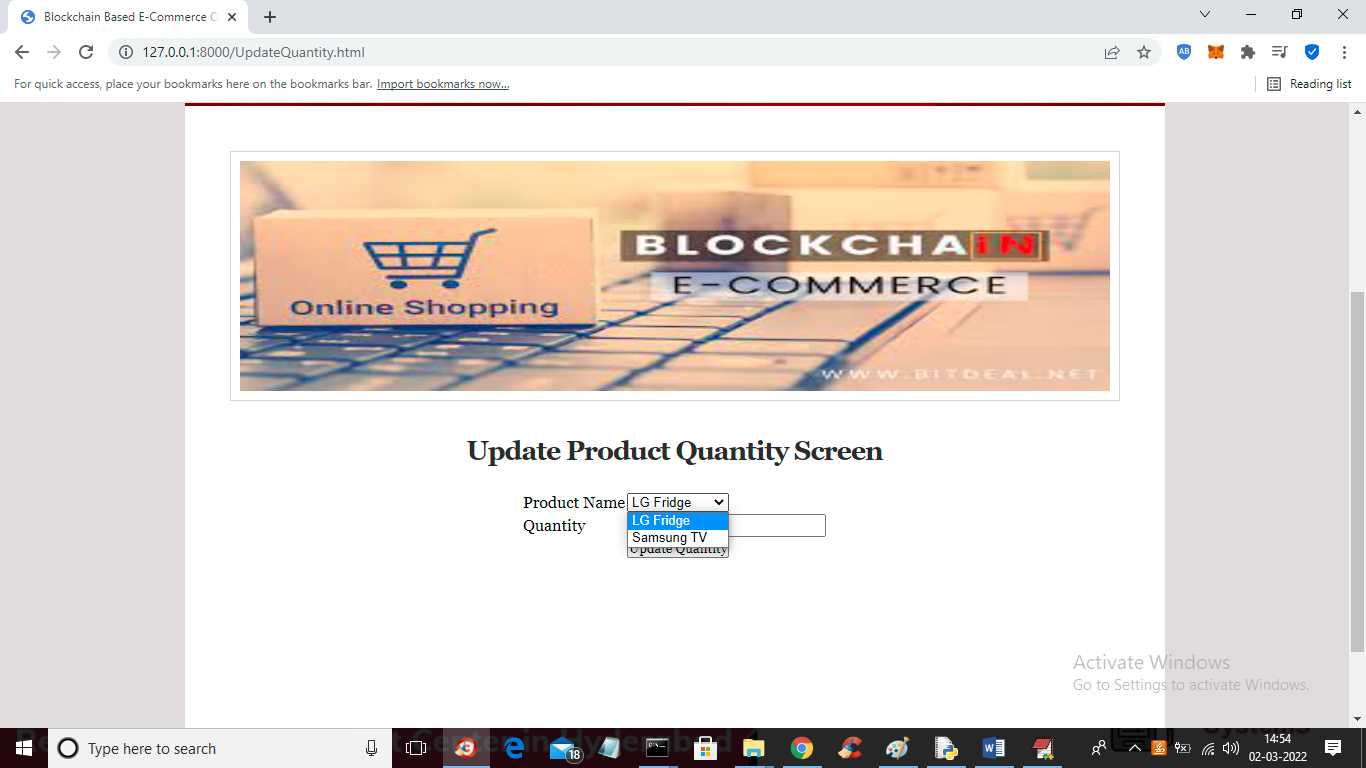
**Figure:-9.6 New Products**

In above screen enter new product details with image and then click on ‘Add Products’ button to add details in Blockchain and get below output



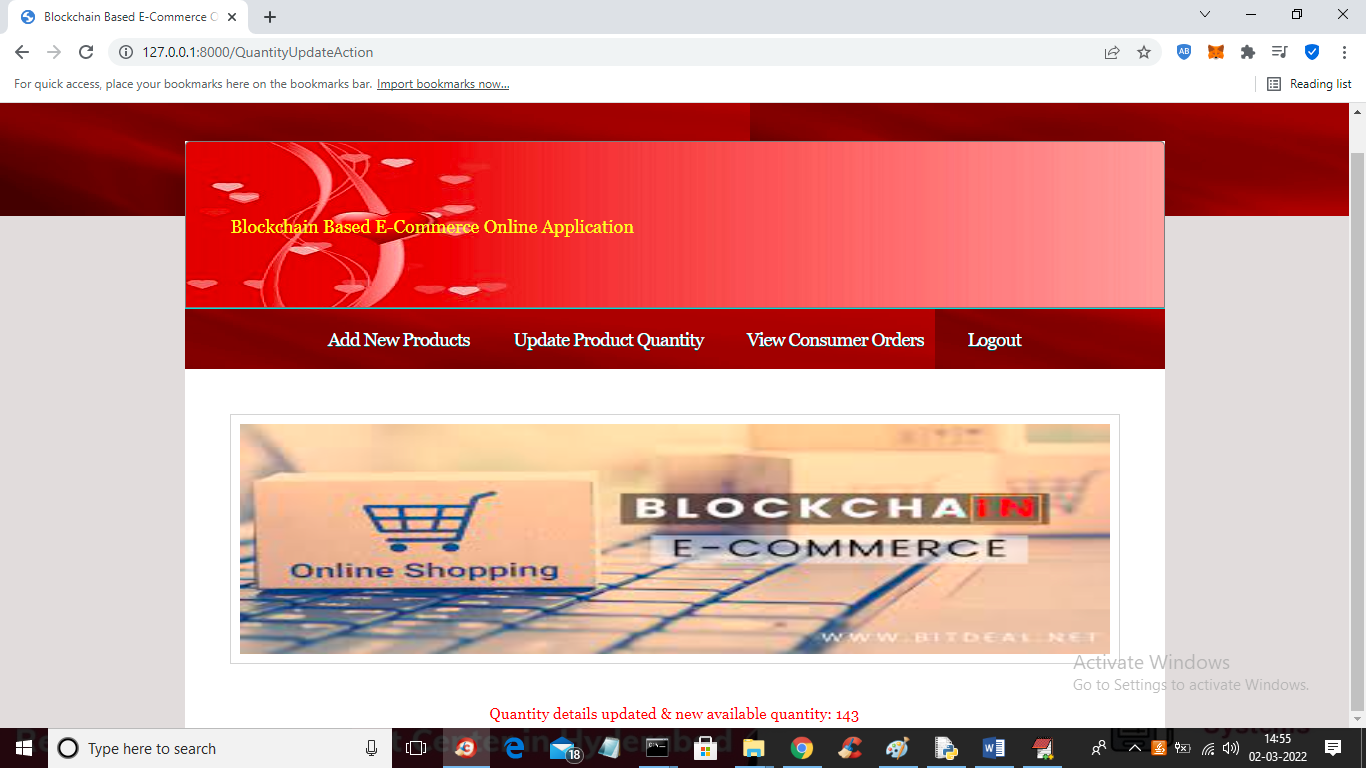
**Figure:-9.7 Products Added**

In above screen in blue color text we can see product details added and we can see hash code of image where image is stored in IPFS. Similarly you can add any number of projects. Now click on ‘Update Product Quantity’ link to update product quantity



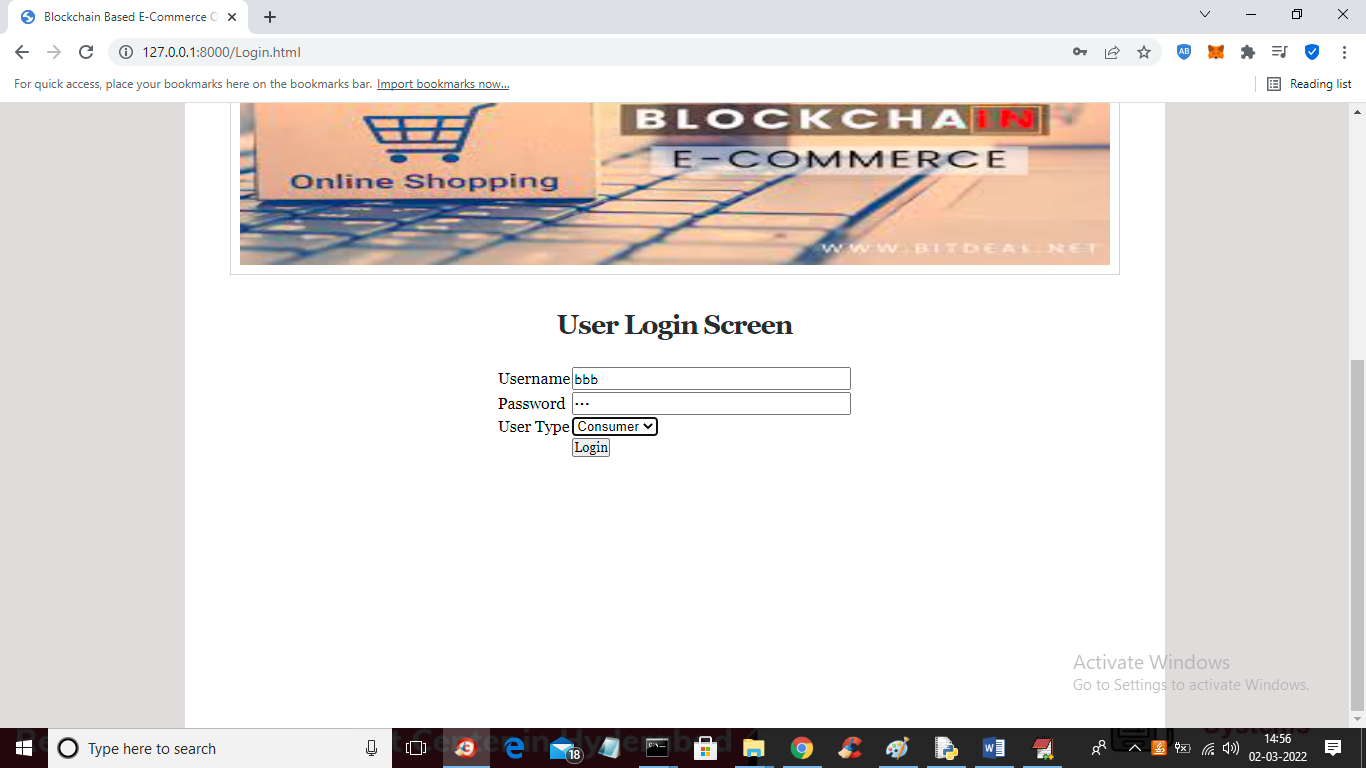
**Figure: - 9.8 Update Product Quantity**

1n above screen select any product name and enter new quantity and press button to get below output



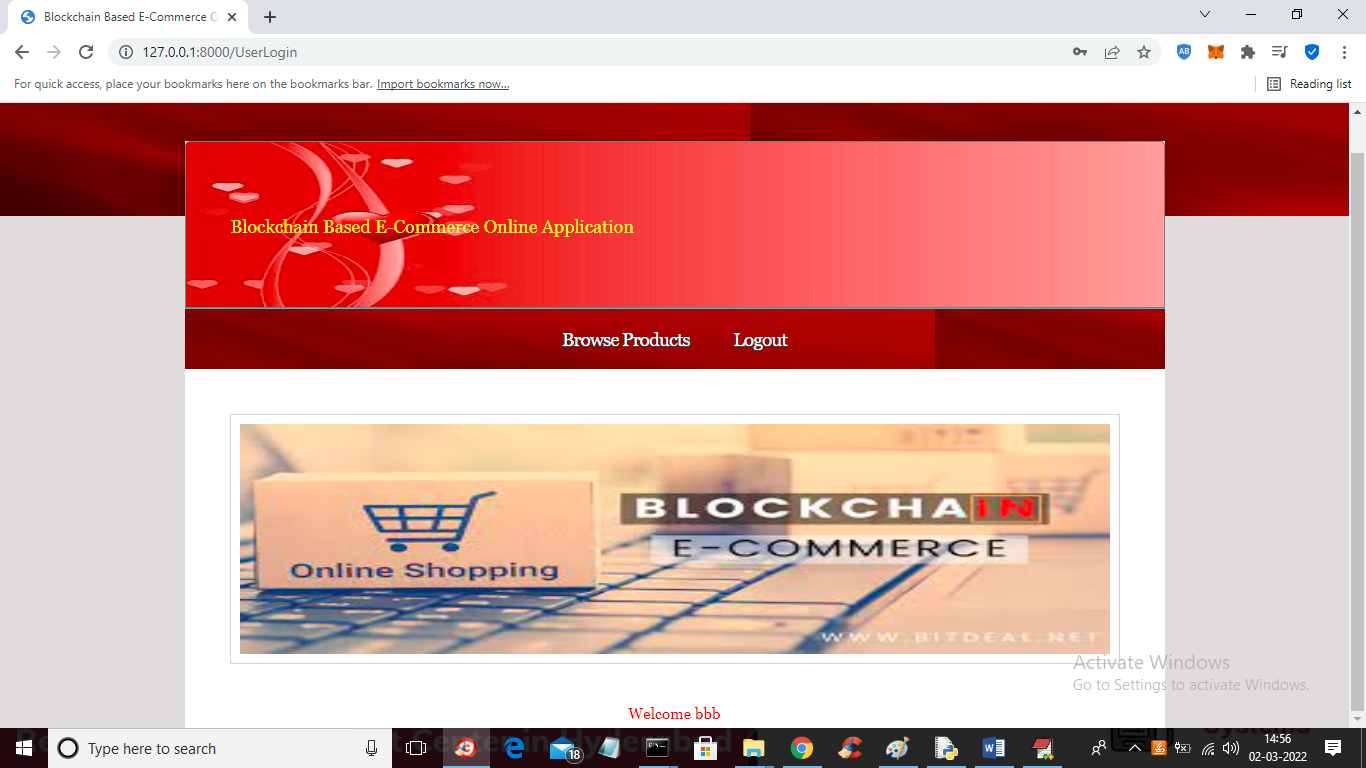
**Figure:-9.9 Quantity Updated**

In above screen in red color we can see quantity is updated and we can see available quantity and now logout and login as customer to purchase products’



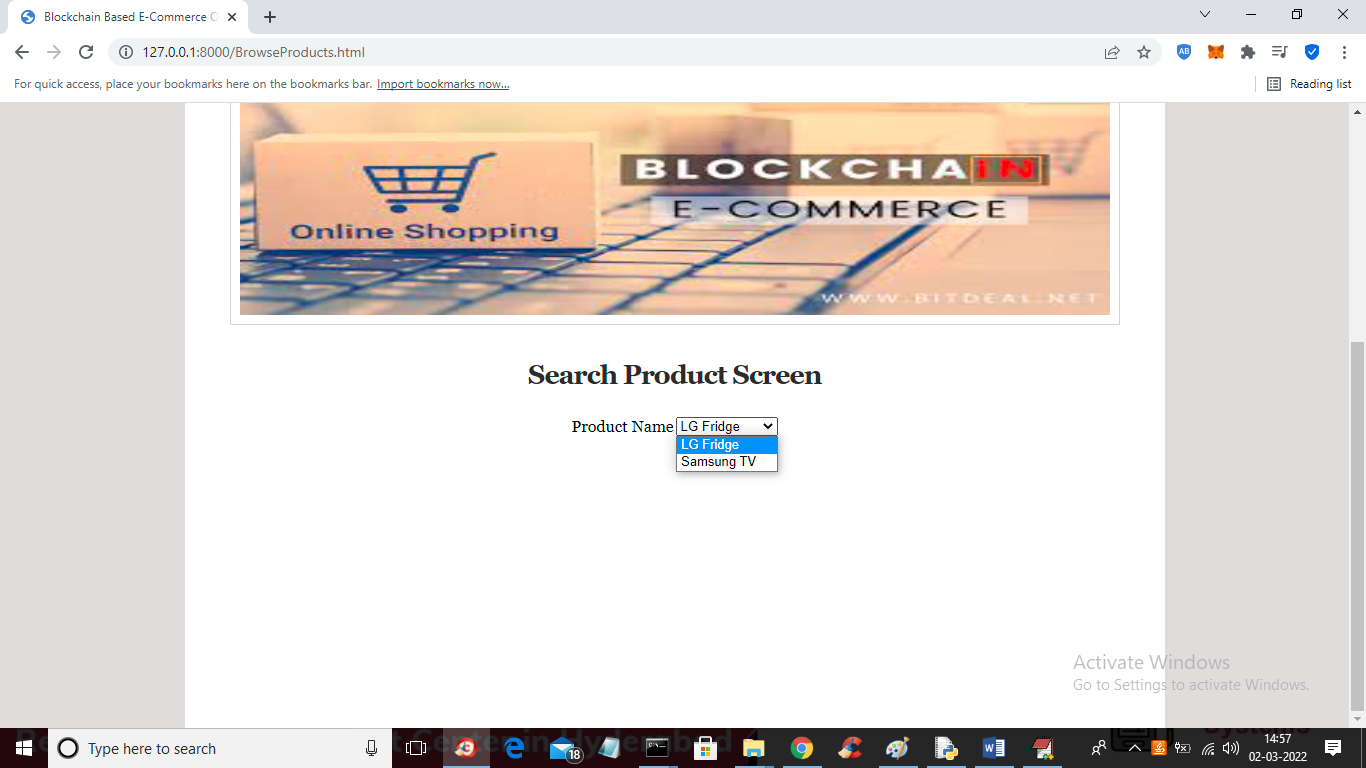
**Figure:-9.10 Customer Login**

In above screen customer is login and after login will get below screen



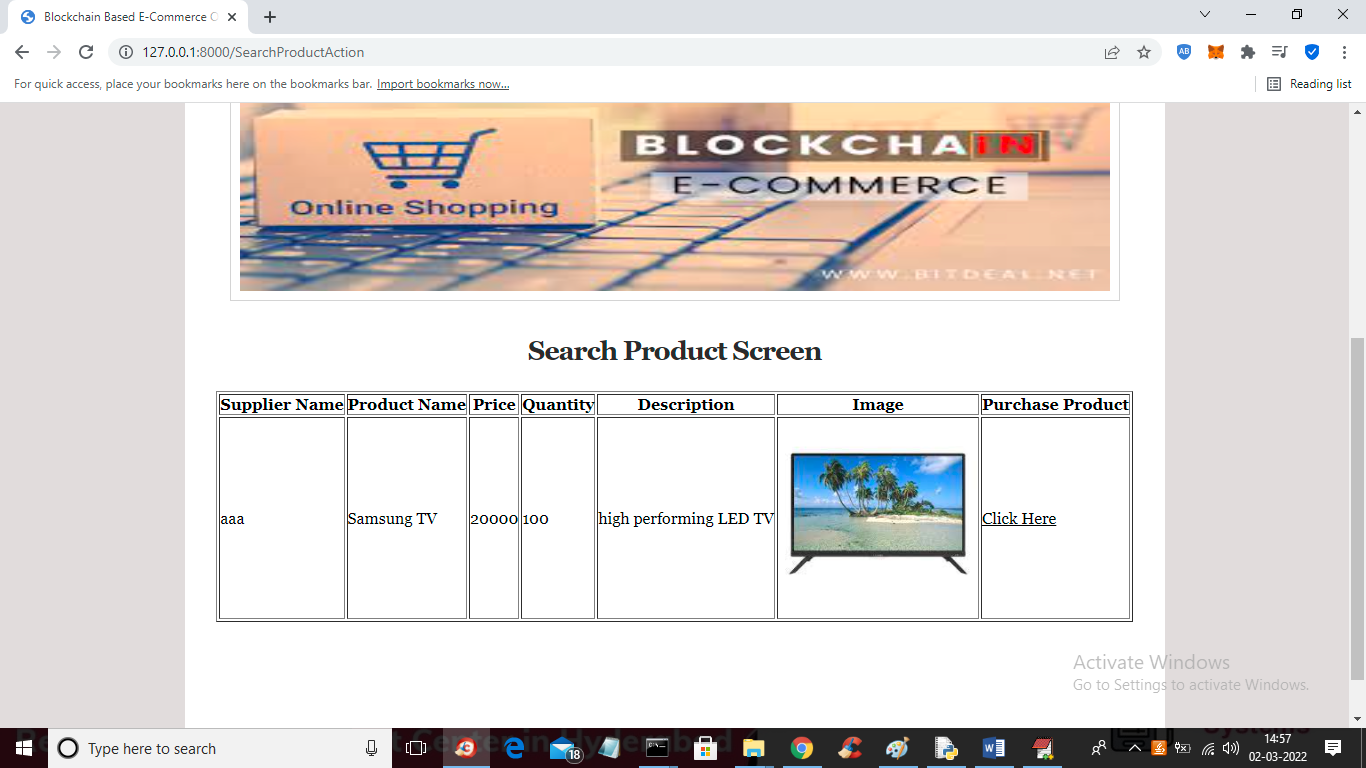
**Figure:-9.11 Browse Products**

In above screen click on ‘Browse Products’ link to get list of products



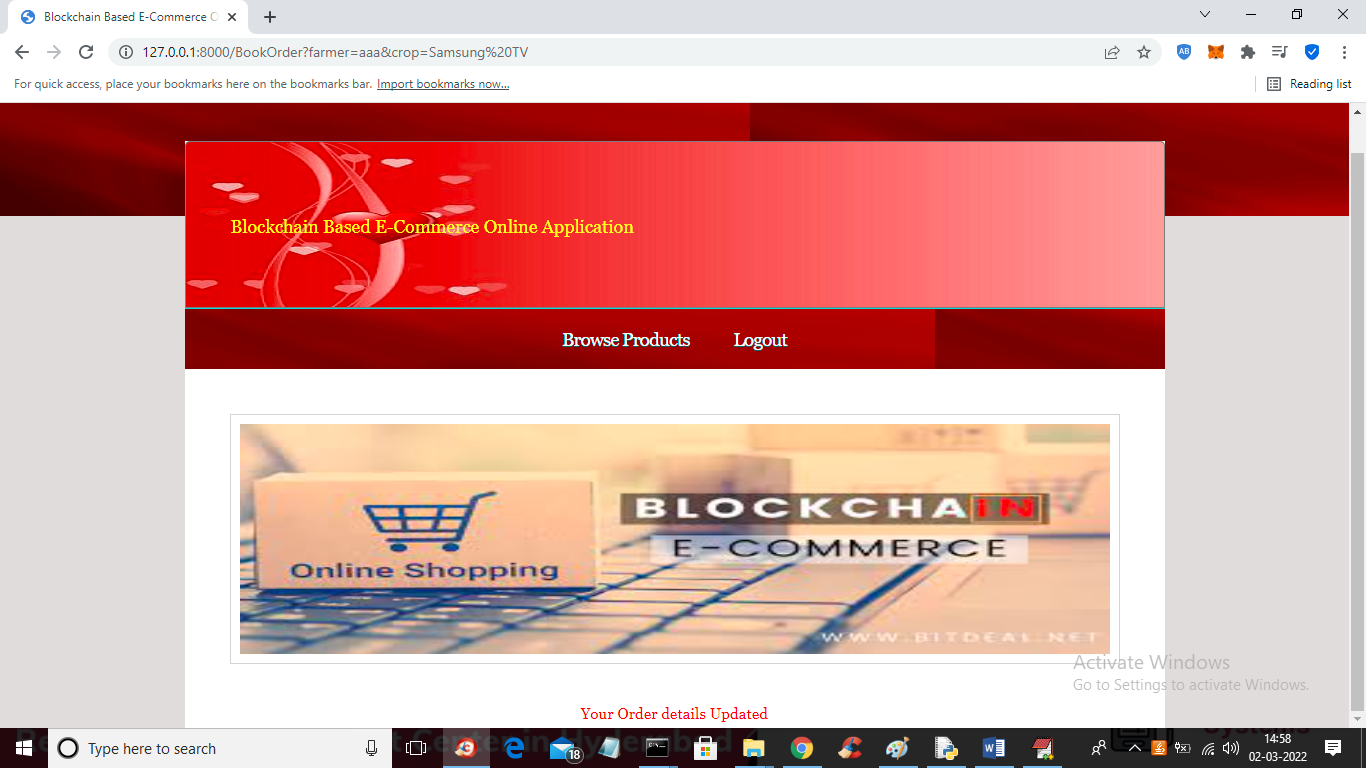
**Figure:-9.12 Search Products**

In above screen user can select desired product and press button to get below list of products



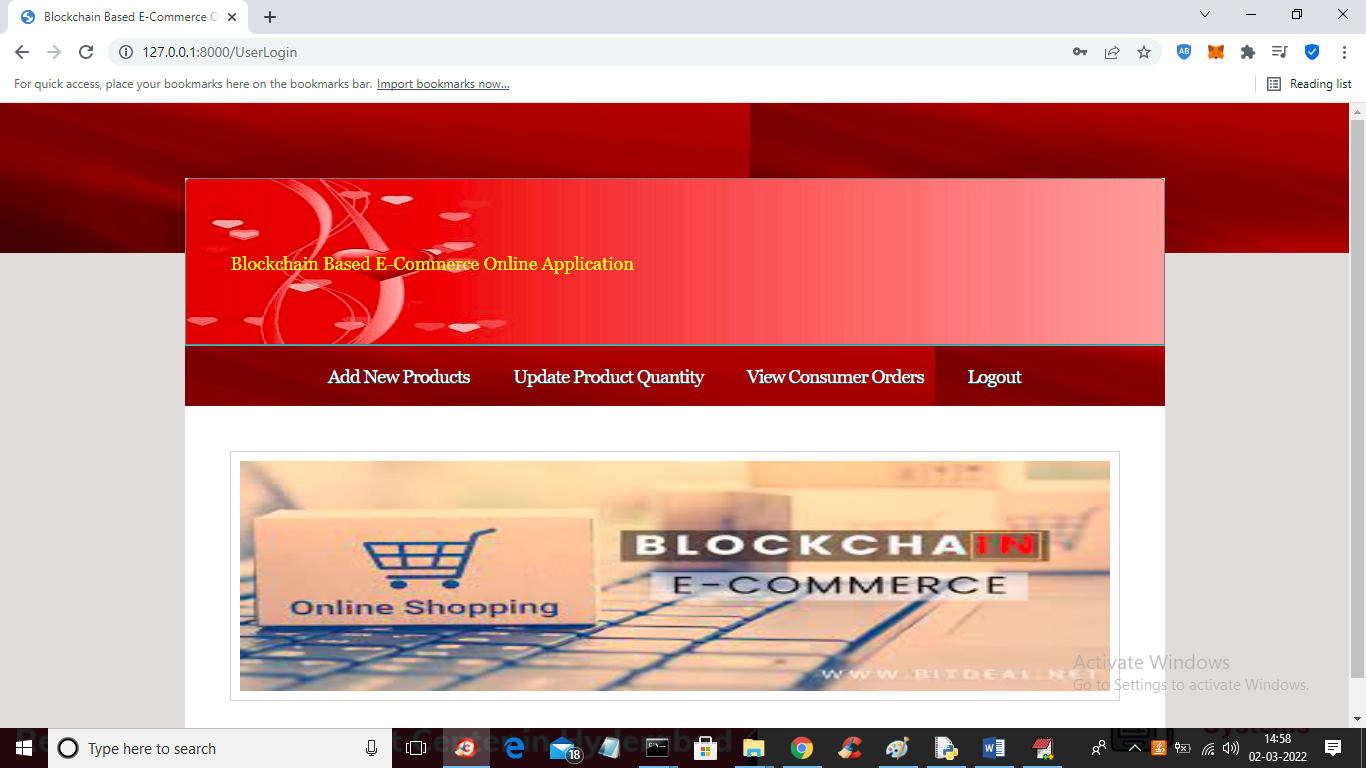
**Figure:-9.13 View Products**

In above screen user can view list of products and then click on ‘Click Here’ link to make an order of this project



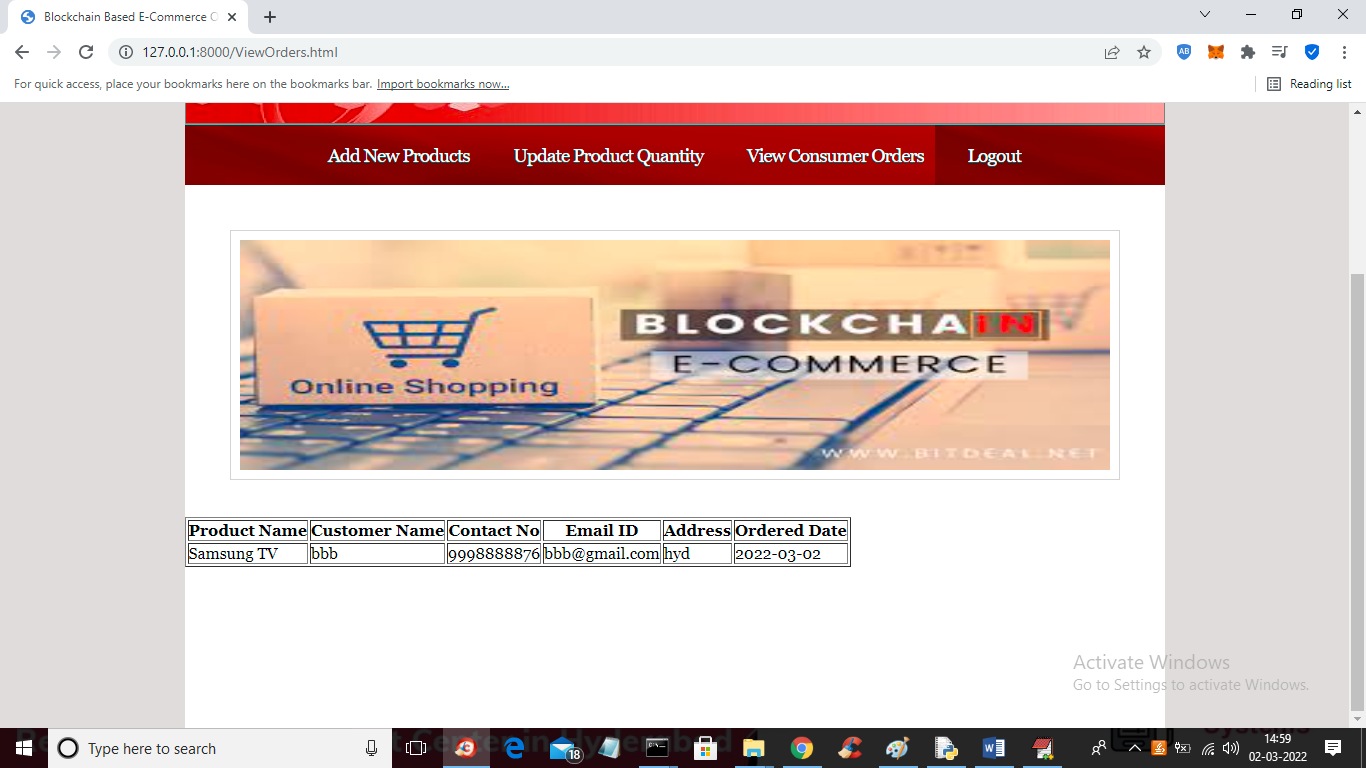
**Figure:-9.14 Place Order**

In above screen in red color text we can see order is updated and now logout and login as supplier to view that order



**Figure:-9.15 View Customer Order**

In above screen Supplier get back to the section and click on ‘View Customer Orders’ link to get below order details



**Figure:-9.16 View Orders Details**

In above screen supplier can see customer contact number and address and complete product delivery

Similarly you can add products and make purchase any number of times

**CHAPTER 10**

**CONCLUSION**

This project successfully implemented Blockchain based architecture for an e-commerce online application aimed at addressing key limitations like data privacy, security, and high availability in traditional centralized e-commerce platforms. The use of Blockchain technology ensured secure, immutable, and transparent recording of all transactions across a decentralized network of nodes. This prevents unauthorized tampering and provides uninterrupted services even if some nodes fail, through redundant data availability.

Additionally, by leveraging smart contracts on the Ethereum network for order and payment processing, the application automated these workflows thereby reducing reliance on intermediaries. Storing product images on IPFS rather than directly on the Blockchain further enhanced decentralization. The project used Python for application logic by integrating frameworks like Truffle and Web3 to interact with the Blockchain ecosystem.

Through a combination of technologies including Blockchain, smart contracts, and IPFS; the developed solution demonstrated core capabilities like user authentication, product cataloguing, shopping cart management and order tracking. Comparative benchmarking indicated improved performance metrics across key parameters such as transaction throughput, latency, and immutability over existing systems.

While this project successfully proved the concept, additional efforts around encrypting sensitive user data, optimizing gas costs of transactions, and supporting Ethereum scaling solutions can further extend capabilities. Overall, the Blockchain based decentralized approach shows strong potential to address rising data security, privacy and resiliency requirements for next generation e-commerce platforms as they continue to serve larger user bases.

**CHAPTER 11**

**REFERENCES**

1. Haoyang,WuL,GuaneshunLil and junhua Wul, “Efficient and Secure Consensus and Privacy Protection Scheme for E-Commerce Consortium Blockchain” https://doi.org/10.1007/s14042-022-13456
2. Mohammad Monirujjaman Khan , Tasneem RoJa,Faris A. Almalki and Maha Aljohani ovel, “Revolutionizing E-Commerce Using Blockchain Technology and Implementing Smart Contracts”. DOI: https://doi.org/10.1155/2022/2213336
3. Gulshan Kumara,∗ , Rahul Sahaa , William J Buchananb , G. Geethaa , Reji Thomasa , Tai-Hoon Kimc and Mamoun Alazabd **“**Decentralized Accessibility of e-commerce Products through Blockchain Technolog “ DOI: https://doi.org/10.1016/j.scs.2020.102361
4. Yanjie Zhu, ”Research on Real-Time Tracking Algorithm of E-Commerce Logistics Information Based on Blockchain Technology”DOI: https://doi.org/10.1007/s 10.1155/2022/7006506.
5. https://www.researchgate.net/publication/334371459\_”Blockchain\_Technologies\_in\_E-commerce\_Social\_Shopping\_and\_Loyalty\_Program\_Applications”
6. Mohammed Huzaif, Mrs. Swathi, “Research on Decentralized E-Commerce Using Blockchain” DOI : https://www.doi.org/10.56726/IRJMETS32178