International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Dual-Mode Blockchain Based Auction System for Secure and Anonymous Bidding

Vaishnavi K, Santhiya S, Ashvitha S, Anusha D
Bachelor Of Engineering, Computer Science and Engineering Nelliandavar Institute Of Technology
Pudhupalayam

Abstract - Traditional selling systems often limit products to local markets and rely heavily on intermediaries, resulting in reduced profit margins, inconsistent quality, and limited market reach. Maintaining consistent quality and ensuring market transparency remain significant challenges in these legacy frameworks. To address these issues, this project proposes a secure and efficient Double Auction System for multi-category product trading.

To enhance security, privacy, and trust, the project integrates advanced cryptographic mechanisms. zk-SNARKs (Zero-Knowledge Succinct Non- Interactive Arguments of Knowledge) are employed for sealed bidding, ensuring that both bidder identities and bid values remain hidden while maintaining mathematical verifiability. Conversely, Linked Ring Signatures are used for open bidding, allowing bid values to remain transparent while masking the identities of the bidders.

A Commit-Reveal Scheme is implemented to prevent bid manipulation and ensure fairness during the submission phase. Additionally, a Reputation Score Algorithm incentivizes honest participation by rewarding users with a trust score based on their historical behavior. Finally, Blockchain technology is integrated via a private blockchain to record all auction data and reports in an immutable and tamper-proof manner. This multi-layered approach ensures a fair, secure, and sustainable trading ecosystem, benefiting both producers and buyers across diverse sectors.

Keywords: Blockchain Auction, zk-SNARKs, Linked Ring Signature, Sealed Bidding, Open Bidding, Reputation Score

CHAPTER 1 - INTRODUCTION

1.1 OVERVIEW

Sellers today have access to a variety of market channels to sell products such as agricultural goods, home appliances, artworks, livestock, and more. While direct-to-consumer methods like sellers' markets or subscription-based models (e.g., community-supported trading) often fetch higher prices, they also demand significant time, effort, and marketing investment.

Wholesale distribution to restaurants, retail stores, or institutions can be profitable; however, managing availability lists, custom orders, and delivery logistics adds complexity—especially for sellers with limited access to modern technologies. The ever-changing technological requirements needed to manage these processes and remain competitive can be a barrier for sellers dealing in agricultural products, home appliances, painting works, livestock, and more.

In contrast, product auctions offer a centralized and efficient marketplace. Through a common platform, sellers can reach a wide pool of buyers across multiple sectors without needing to manually handle negotiations, marketing, or order customization. This allows them to focus on their core work such as agriculture, production, or artistic creation, while buyers benefit from real-time visibility into seasonal availability, product quality, and competitive pricing.

Buyers from various sectors—agriculture, retail, artworks, institutions, and more—can inspect products, assess quality, and place real-time bids. Auctions enable cost savings during high-supply periods and create opportunities to source premium goods by bidding in real-time ahead of traditional market cycles. This system enables direct interaction between producers and procurers through a real- time auction platform, eliminating middlemen.

Online and live auctions support both open bidding and sealed bidding environments. In open bidding, participants can view other bids, encouraging competitive offers while maintaining anonymity. In sealed bidding, both the bids and bidder identities remain private until the reveal stage, ensuring fairness and strategic decision-making.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Ultimately, auctions benefit both buyers and sellers by simplifying procurement, reducing overhead, and supporting high-volume trade across diverse product categories.

1.1.1 Types of Auctions

In a typical auction, the auctioneer announces the price and other required details of the product and conducts the event, where buyers or bidders submit the maximum amount they are willing to pay for the item or service. The highest bidder wins the auction and obtains possession of the item or service when the auction is initiated by producers. This is known as a forward auction, where bids increase during the auction.

On the other hand, when the auction is initiated by procurers, the lowest bid value is selected as the winner. This type is known as a reverse auction.

Some common examples of forward auctions include:

- **Public Auctions** – An auction where the seller is a government entity and the buyers are the general public.
- **Private Auctions** – Auctions conducted privately when the sellers and buyers know each other.
- **Live Auctions** – Open auctions where potential bidders participate live, either physically or virtually.
- **First Lot Auctions** – A single lot is sold to the highest bidder simultaneously.
- **Sealed Bid Auctions** – Buyers submit their bids privately in sealed envelopes.
- **Sale in Lots** – Goods are sold collectively as a batch or lot.
- **Auction Sale of Unsold Lots** – Unsold items from previous auctions are offered again for sale.

1.2 PROBLEM STATEMENT

The current trading system for products from diverse sectors such as agricultural goods, livestock, artworks, and other commodities faces several critical challenges that hinder efficiency, transparency, and trust.

Traditional auction platforms often lack visibility into bid values and bidder identities, which can lead to unfair practices, bias, and market manipulation. The absence of robust privacy mechanisms compromises participant confidentiality and may result in distrust among stakeholders.

Bid manipulation remains a major issue due to the lack of secure methods to validate and timestamp bids. Additionally, many existing systems do not support real-time tracking or dynamic decision-making, which slows down the bidding process and reduces market responsiveness.

For perishable and time-sensitive products such as fresh produce and livestock, delays in auction outcomes can lead to spoilage and financial losses. Furthermore, the absence of a reputation scoring mechanism makes it difficult for participants to evaluate the credibility of buyers or sellers, leading to unreliable transactions.

Producers, who are often confined to local markets, face restricted reach and reduced profit potential. The lack of integrated and secure payment systems also contributes to transaction delays and disputes, creating friction in the trading process.

To overcome these limitations, the proposed system integrates blockchain technology with advanced cryptographic mechanisms such as zk-SNARKs for sealed bidding, Linked Ring Signatures for open bidding, and a Reputation Score Algorithm. This combination ensures privacy, fairness, transparency, and traceability in a decentralized marketplace, creating a secure, scalable, and efficient auction ecosystem for multi-category product trading.

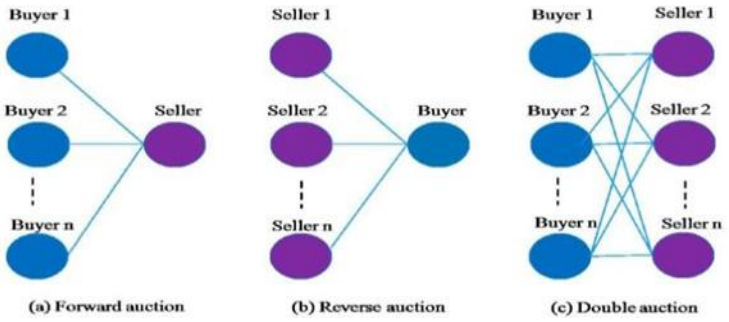
1.3 DOUBLE AUCTION

A double auction is a powerful and widely used trading mechanism that facilitates transactions between multiple buyers and multiple sellers. In this system, Sellers submit ask prices for the goods or services they wish to offer while Buyers submit bid

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

prices indicating the maximum price they are willing to pay. The auction mechanism then determines a clearing price (p) that matches supply and demand: Sellers whose ask prices are less than or equal to *p* complete a sale. Buyers whose bid prices are greater than or equal to p complete a purchase. This format ensures fair allocation and price discovery through market-driven participation. A real-world example of a double auction is the stock exchange, where multiple orders from buyers and sellers are matched based on bid-ask prices.

Fig 1.3 Representation of different auctions



A double auction effectively combines the features of both forward and reverse auctions, enabling flexible market participation. One notable variant is the Walrasian auction (Walrasian tâtonnement), where an auctioneer collects bids from both buyers and sellers and continuously adjusts the price to reach equilibrium—a point where supply equals demand. The auction concludes once this market-clearing price is reached.

Another type is the Barter Double Auction, where monetary exchange is absent. Instead, participants present multi-attribute offers and demands, and trades occur based on mutual compatibility. In such cases, Euclidean distance is often used for mathematical modeling of satisfaction levels by representing offers and demands as vectors.

Overall, double auctions provide a fair, decentralized, and efficient mechanism for market-based resource allocation, making them ideal for systems involving diverse buyers, sellers, and product categories.

1.3.1 BLOCKCHAIN TECHNOLOGY

Blockchain technology is a structure that stores transactional records, known as blocks, across multiple databases called a chain, within a network connected through peer-to-peer nodes. This storage system is commonly referred to as a digital ledger.Every transaction recorded in the ledger is authenticated using the digital signature of the owner, ensuring that the transaction is genuine and protected from tampering. As a result, the information stored within the digital ledger is highly secure.

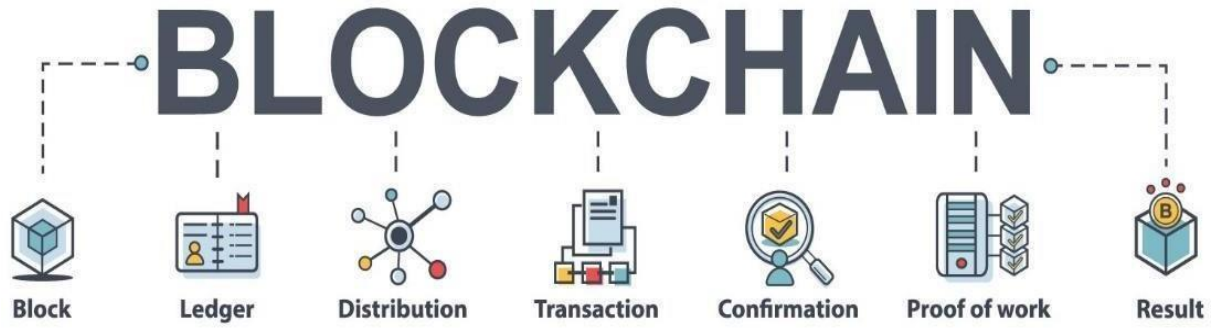


Fig 1.3.1 Blockchain Overview

Security is not the only advantage of blockchain technology. Blockchain also ensures that the owner of the data remains

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

anonymous. Sensitive information related to auctions and transactions is securely recorded in the system. Whenever a user attempts to access or verify data, the entire chain is validated to detect any alterations. If any tampering is detected, the responsible node or miner can be removed from the network. From large technology companies to startups, many organizations are adopting blockchain-based cloud storage solutions to digitally transform their operations. Blockchain offers a secure and cost-effective storage solution because multiple organizations contribute computing power and storage space collectively. This collaborative model reduces cloud storage costs, while contributors are rewarded for sharing their resources.

1.4 AIM AND OBJECTIVE

Aim

The aim of this project is to design and develop a secure, transparent, and efficient dual-mode auction system for trading a wide range of products, including agricultural goods, livestock, artworks, and home appliances.

The system leverages cryptographic techniques such as zk-SNARKs for sealed bidding and Linked Ring Signatures for open bidding, ensuring bid privacy and participant anonymity. Additionally, blockchain technology is integrated to

immutably record auction data, enhancing trust, preventing manipulation, and enabling fair and decentralized market participation.

Objectives

The main objectives of the project are:

- To develop a dual-mode auction system supporting both sealed and open bidding formats.
- To ensure bid privacy using zk-SNARKs (Zero-Knowledge Succinct Non- Interactive Argument of Knowledge).
- To mask bidder identities using Linked Ring Signatures in open bidding.
- To establish a Reputation Score Algorithm that promotes honest participation and trust.

1.5 SCOPE OF THE PROJECT

This project focuses on developing a secure, privacy-preserving, and transparent dual-mode auction system for trading diverse products, including agricultural goods, livestock, home appliances, and artworks. The key components of the project include:

- Auction Types – Supports both sealed and open bidding mechanisms, ensuring bidder confidentiality using zk-SNARKs and Linked Ring Signatures.
- Blockchain Integration – Records all transactions on a tamper-proof ledger, ensuring transparency and immutability.
- Reputation Scoring – Evaluates the credibility of participants based on historical bidding behavior and auction participation.
- Commit-Reveal Scheme – Prevents bid manipulation by verifying the authenticity of sealed bids through a two-phase verification process.
- Payment System – Enables secure, fast, and direct payments between producers and procurers through integrated payment gateways or QR- based systems.
- Real-Time Monitoring – Provides dashboards for users and administrators to track auction status, bid activity, and outcomes.
- User Interface – Offers intuitive and user-friendly interfaces designed for easy access by producers, procurers, and administrators.

Overall, this project aims to significantly improve the efficiency, fairness, security, and scalability of auction-based

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

trading across multi-category markets.

CHAPTER 2 - LITERATURE SURVEY

2.1 TITLE: A Safe and Secure Online System for Bidding Using Blockchain Technology

AUTHORS: Dr. Anuradha S. G, H. Vadiraja

DESCRIPTION: The authors propose the use of zk-SNARKs for sealed bidding and Ring Signatures to protect participant identities. The proposed cryptographic techniques enhance both the privacy and verifiability of the bidding process. Bids remain confidential while the correctness of transactions can be mathematically verified without revealing sensitive bid information. The study addresses privacy concerns in online auctions and improves the security of the bidding process. However, the system does not include mechanisms such as linkable anonymity, reputation evaluation, or commit–reveal schemes, which are important for preventing fraudulent activities and ensuring fairness in large-scale auction environments.

2.2 TITLE: Secure Online Auction System AUTHORS: S. Rajesh, K. Venkatesh

DESCRIPTION: Secure Online Auction System incorporates cryptographic techniques such as AES-128 and RSA algorithms to ensure that user credentials and bid data remain secure. Encryption mechanisms are used to protect the system from unauthorized access and provide a safe environment for online auction transactions. The system ensures that sensitive information is encrypted before storage or transmission. However, the auction platform operates in a centralized architecture using a MySQL database, which limits transparency and trust among participants. Furthermore, the system does not provide privacy protection for bidders and lacks mechanisms to prevent fake users and bid manipulation.

2.3 TITLE: Online Auction System Based on Distributed Technology AUTHORS: Hisham S. Galal, Amr M. Youssef

DESCRIPTION: Online Auction System Based on Distributed Technology presents an early implementation of a blockchain-based auction system that utilizes SHA-256 hashing to maintain data integrity. The authors emphasize the importance of eliminating centralized authorities in auction platforms by using distributed ledger technology. In this system, bid data is stored on the blockchain in an immutable format, which ensures tamper-proof records and transparent verification of transactions. Although the decentralized architecture improves system transparency and trust, the approach lacks privacy-preserving mechanisms such as bidder identity protection and sealed bidding features. As a result, the system remains vulnerable to information exposure and identity leakage.

2.4 TITLE: Novel Double Auction Mechanisms for Agricultural Supply Chain Trading

AUTHORS: Yifan Zhang, Shouyang Wang, Xiaojie Li

DESCRIPTION: Novel Double Auction Mechanisms for Agricultural Supply Chain Trading research addresses inefficiencies in Agricultural Supply Chain Trading (ASCT), including high transaction costs, product spoilage, and complicated bidding processes. The authors propose two novel Multi-Unit Double Auction mechanisms known as MNR and MTR to improve resource allocation in both oversupply and overdemand scenarios. These mechanisms enhance pricing efficiency and reduce carbon emissions in agricultural trading, as demonstrated through numerical simulations. While the proposed approach improves profitability and market efficiency, it primarily focuses on single- product and price-only auctions. Future improvements may include multi- attribute bidding, multi-round auction strategies, and integration of logistics considerations to improve real-world applicability.

2.5 TITLE: Comprehensive Survey on Auction Mechanism Design for Cloud/Edge Resource Management and Pricing

AUTHORS: Nafiseh Sharghivand, Farnaz Derakhshan

DESCRIPTION: Comprehensive Survey on Auction Mechanism Design for Cloud/Edge Resource Management and Pricing provides a comprehensive survey of auction-based mechanisms used for resource management and pricing in cloud and edge computing environments. The paper discusses challenges such as heterogeneous resource requirements, dynamic user demands,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

and the need to align incentives between service providers and consumers. The authors classify auction mechanisms based on several factors, including bid direction (forward, reverse, and double auctions), resource heterogeneity, and auction environments such as offline, online, and sequential auctions. The survey also explores the role of multi-attribute bidding and complex resource allocation models in cloud and edge platforms. The study concludes that further research is required to develop more efficient auction mechanisms that better address real-world computing demands.

2.6 TITLE: A Blockchain-Based Framework for Secure Public and Sealed- Bid Auctions

AUTHORS: Hisham S. Galal, Amr M. Youssef

DESCRIPTION: A Blockchain-Based Framework for Secure Public and Sealed-Bid Auctions proposes a hybrid auction framework that integrates both open and sealed bidding mechanisms using blockchain technology. The system uses AES encryption and public-key authentication to secure communication between participants. Blockchain serves as a decentralized ledger that records auction events and maintains transparency and tamper-proof data storage. While the framework successfully ensures bid security and decentralized data management, it does not incorporate advanced privacy-preserving techniques such as zk-SNARKs. Additionally, the system does not consider behavior-based trust mechanisms such as reputation scoring or commitment-based bid verification.

2.7 TITLE: AStERISK: Auction-Based Shared Economy Resolution Markets

AUTHORS: Benedikt Bünz, Ben Fisch, Alan Szepieniec

DESCRIPTION: AStERISK presents a privacy-preserving sealed-bid auction model designed for shared economy platforms. The system utilizes zk-SNARKs to ensure strong bidder anonymity while allowing verification of bid correctness without revealing the actual bid values. Blockchain technology is used to securely record auction transactions, ensuring transparency and tamper-proof logs. Although the model effectively ensures privacy in sealed bidding environments, it does not support open bidding mechanisms. Furthermore, it does not provide solutions for issues such as multi-bidder fraud detection, duplicate participation, or reputation management in decentralized marketplaces.

2.8 TITLE: Trustworthy Sealed-Bid Auction with Low Communication Cost Atop Blockchain

AUTHORS: Qian Zhang, Yong Yu

DESCRIPTION: Trustworthy Sealed-Bid Auction with Low Communication Cost Atop Blockchain study proposes a sealed-bid auction protocol that utilizes homomorphic encryption and mix-and-match techniques to compute auction results securely over encrypted data. The system supports public verifiability through zero-knowledge proofs while significantly reducing communication overhead between participants. Designed for blockchain deployment, the protocol ensures secure storage of auction records and improves system scalability. However, the system relies on a trusted cryptographic setup and may experience performance overhead due to the extensive use of encryption and verification mechanisms during the auction process.

CHAPTER 3 - SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

Processor	:	Dual-core (Intel Core i5 / AMD Ryzen 5).
Memory	:	8GB RAM for smooth performance.
Storage	:	500GB SSD for fast data access.

3.2 SOFTWARE SPECIFICATION

Operating System	:	Windows 10 / 11 for development
Programming	:	Python 3.13 as the core language.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Web Server	:	Apache HTTP Server for hosting.
Database	:	MySQL 8.0 for data management.
Framework	:	Flask 2.0 for application logic.
Web Design	:	HTML5, CSS3, JavaScript, Bootstrap5.
Tools	:	PyCharm for development and deployment.

CHAPTER 4 - ANALYSIS OF THE PROJECT

4.1 USE CASE DIAGRAM

The Use Case Diagram outlines the primary interactions between users (Producers, Procurers from various sectors, Admin) and the system. It includes use cases such as Register/Login, Announce Auction, Place Bid, View Auction, Determine Winner, and Make Payment as shown in Fig 4.1. This diagram highlights the system's functional boundaries and user roles.

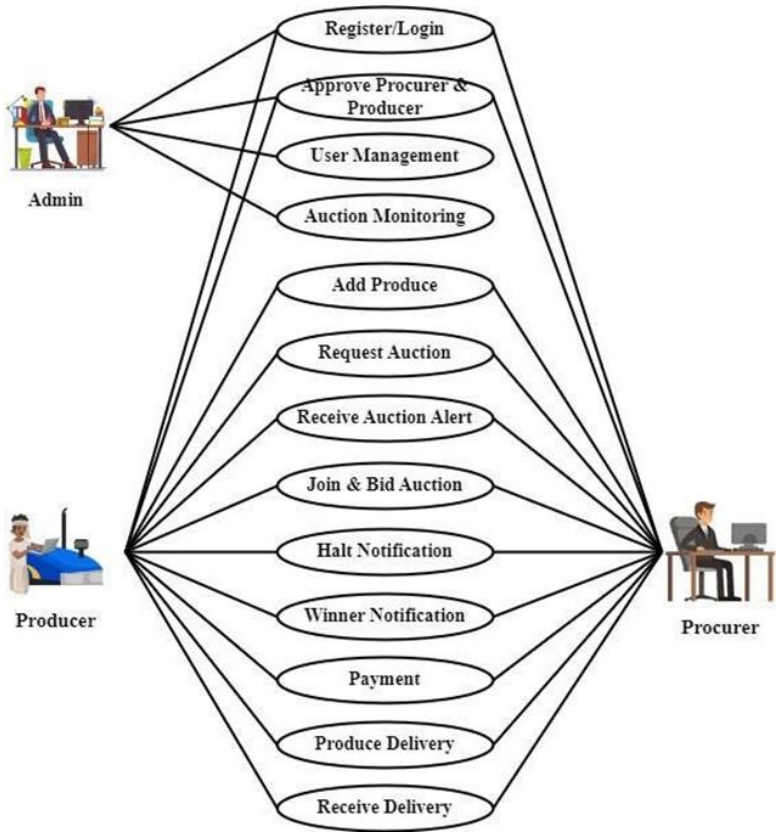


Fig 4.1 Use Case Diagram

4.2 ACTIVITY DIAGRAM

The Activity Diagram describes the workflow of actions in the system, such as how a user creates an auction, places a bid, and processes payments. It shows decisions, parallel processes, and the flow of control, offering a clear view of dynamic behavior as shown in Fig 4.1.4.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

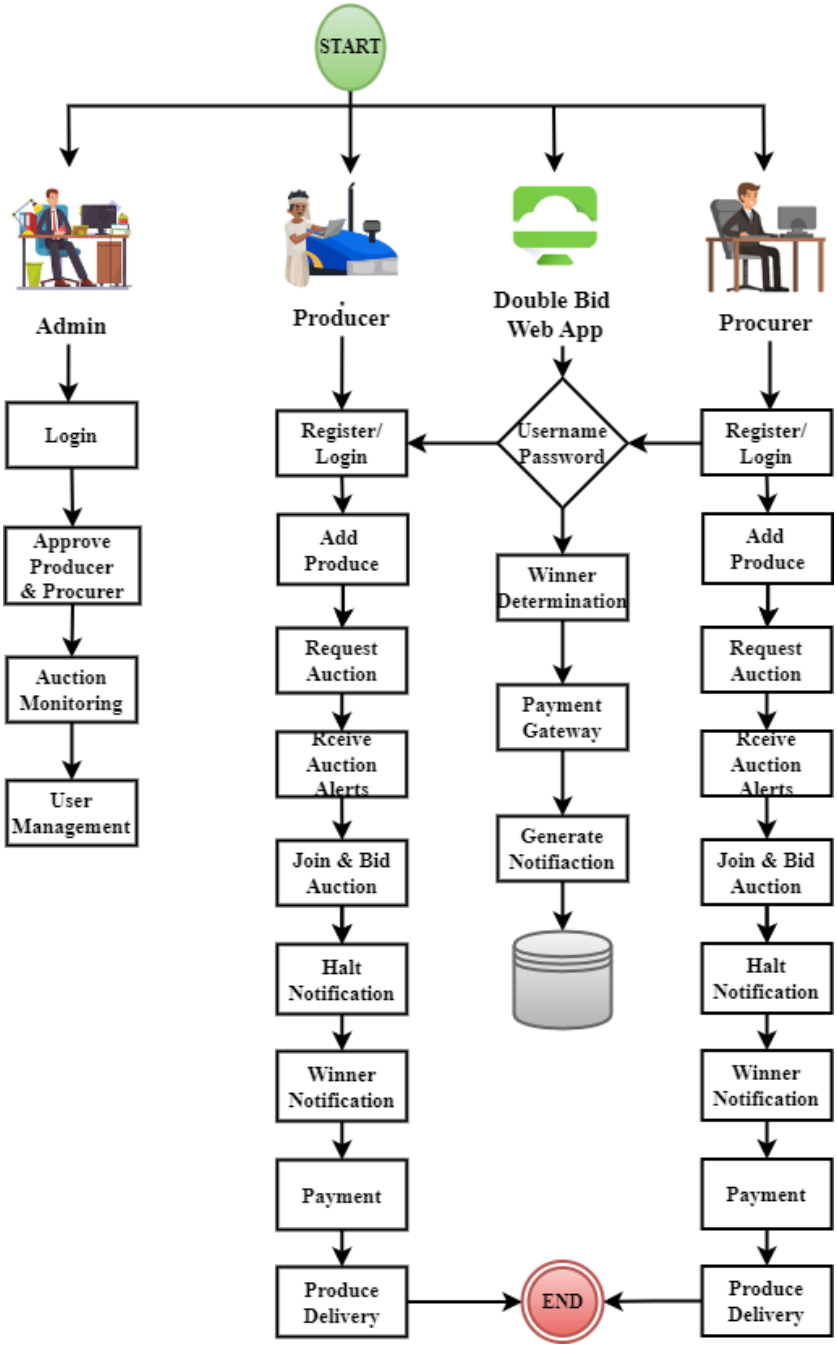


Fig 4.2 Activity Diagram

4.3 USER AUTHENTICATION AND ROLE SELECTION

The workflow of the Dual-Mode Blockchain Auction System begins with user authentication. All participants, whether producers or procurers, must first register and then log in to access the platform. Upon successful login, the system presents the user with a choice of role, directing them down one of two primary paths based on their intent. A producer wishing to sell goods can initiate a forward auction by providing the necessary product details. Conversely, a procurer looking to purchase goods can either browse and join existing producer-initiated auctions or, in a reverse auction scenario, request a new auction for a specific product they require.

4.4. BIDDING MECHANISM AND AUCTION TYPES

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The core of the process is the bidding phase, which bifurcates based on the auction type. For sealed-bid auctions—typically used when procurers request a product to ensure fair and unbiased pricing—the system employs zk-SNARKs to keep both the bidder's identity and the bid amount completely confidential. For open auctions, usually initiated by producers to foster competitive bidding, the system utilizes Linked Ring Signatures. This mechanism allows all participants to observe the rising bid amounts in real time while ensuring that the bidders' identities remain anonymous.

4.5. WINNER DETERMINATION AND SECURE PAYMENT

Once the predefined auction period concludes, the system automatically executes the winner determination logic. This logic selects the winner based on the auction's initiator: the highest bid wins in a producer-initiated auction, while the lowest bid wins in a procurer-initiated auction. Following the winner announcement, the system facilitates a secure payment process between the two parties.

4.6. REPUTATION UPDATE AND BLOCKCHAIN RECORDING

To maintain a trustworthy ecosystem, the reputation scores of both the buyer and seller are updated based on the successful and timely completion of the transaction. Finally, a comprehensive record of the entire auction—including all bids, the winner, and the final transaction—is immutably logged onto a private blockchain. This final step ensures transparency, traceability, and non- repudiation, effectively concluding the auction lifecycle.

CHAPTER 5 - SYSTEM ESIGN

5.1. EXISTING SYSTEM

- **Traditional Market-Based Selling:**Sellers typically sell their products in physical local markets, often under pressure from local demand and seasonal pricing. This method offers limited geographical reach and prevents farmers from accessing competitive buyers, which results in lower profits and poor market efficiency.
- **Middlemen and Distributors :**In this system, producers depend heavily on agents, brokers, or middlemen to sell their goods. These intermediaries often manipulate prices to maximize their own profits, leading to reduced earnings and a lack of transparency for the actual producers.
- **Fixed-Price E-commerce Platforms:**Online platforms enable producers or sellers to list their product at predetermined prices. While convenient, these platforms do not account for real- time fluctuations in supply and demand, thus failing to offer dynamic pricing and potentially undercutting sellers' profits
- **Sealed Bidding Auctions:**These auctions allow participants to submit bids privately, where no one knows the others' bids. While this preserves confidentiality, the lack of transparency may lead to trust issues among participants and disputes over fairness in winner determination.
- **Open Bidding Auctions:**In these auctions, bid values are visible to all participants in real-time, encouraging competitive bidding. However, this openness can result in strategic bid manipulation and the exposure of bidder identities, making the process vulnerable to unethical practices.

5.1.1 Disadvantages

- The limited geographical reach of traditional markets reduces farmers' profit margins and market competitiveness.
- Heavy dependence on middlemen often results in price manipulation and reduced transparency for producers.
- Fixed-price models on e-commerce platforms lack flexibility and fail to adapt to real-time supply and demand dynamics.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Sealed bidding mechanisms, while private, lack transparency and may lead to trust issues and disputes.
- Open bidding compromises participant anonymity, increasing the risk of bid manipulation and collusion.
- Farmers have minimal control over pricing strategies and buyer interactions, weakening their market influence.
- The absence of automation leads to operational inefficiencies, delays, and increased manual intervention.
- There is no robust system to verify the authenticity of bidders or validate the integrity of their bids.
- The lack of secure, immutable transaction records compromises accountability and trust.
- Reputation systems for buyers and sellers are inadequate or entirely absent, affecting credibility.

5.2. PROPOSED DUAL-BLOCKBID

- **Dual Auction Initiation (Seller & Procurer)** The system allows both producers (sellers) and procurers (buyers) to initiate auctions. Sellers can list their other high-value goods or product for sale, and buyers from diverse sectors can open requirements for procurement. This bidirectional model ensures a dynamic and flexible marketplace that caters to diverse trading scenarios.
- **Sealed Bidding using Zk-SNARKs** When procurers from various sectors initiate an auction by requesting for a product, producers from diverse sectors submit sealed bids. The system uses Zero-Knowledge SuccinctB Non- Interactive Argument of Knowledge (Zk-SNARKs) to hide both the bid values and the identities of the bidders. This cryptographic technique ensures confidentiality and eliminates bias during winner selection.
- **Open Bidding using Linked Ring Signatures** When producers initiate auctions, where the procurers place bids where the bid amounts are visible and bidder's identities are hidden using Linked Ring Signature algorithms. This maintains anonymity while allowing transparency in bid amounts, thus reducing manipulation and fear of retaliation.
- **Automated Winner Determination Logic** The system includes smart logic to automatically determine winners. If a producer initiates the auction, the highest bid wins. If a procurer initiates the auction, the lowest bid wins. This ensures a fair and consistent auction strategy that aligns with the interests of both parties
- **Blockchain-Enabled Auction Simulator** The proposed system implements a blockchain-based auction simulator using Python, Flask, MySQL, Bootstrap, and WampServer. It provides a secure and decentralized platform for auctioning multi-category products. By leveraging blockchain, all transactions are recorded r immutably, ensuring transparency, traceability, and trust among stakeholders.
  - Sealed Bidding Mode - Uses zk-SNARKs for complete bid confidentiality
  - Open Bidding Mode – Uses Linked Ring Signatures for anonymous yet transparent bidding
  - Commit-Reveal Scheme for Fairness-Bids cannot be changed after submission No last-minute manipulation Transparent winner declaration
  - This dual-mode architecture makes the system adaptable to various trading environments while eliminating intermediaries and ensuring tamper-proof auction processes.
- **Reputation Scoring System** To promote accountability and trust, the system implements a reputation scoring mechanism based on successful transactions, timely participation, and reliability. Participants with higher scores are prioritized in future auctions, thus rewarding good behavior and encouraging transparency, which may overheads the existing system as listed in table 5.3.

5.2.1. Benefits of the Proposed System

- Eliminates intermediaries, ensuring fair prices for sellers and buyers.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Enhances transparency and trust through blockchain-recorded auction data.
- Protects bidder identity and bid value using advanced cryptographic algorithms (Zk-SNARKs, Linked Ring Signatures).
- Automates winner selection, reducing human intervention and bias.
- Improves reliability through a reputation-based scoring system.
- Secures payments with integrated digital payment processing.
- Allows flexible auction creation by both producers and procurers.
- Supports multimedia listings (images, videos) for better product visibility.
- Prevents bid manipulation using commit-reveal scheme.
- Facilitates real-time bidding and dynamic pricing based on demand and supply.

5.3 System Architecture

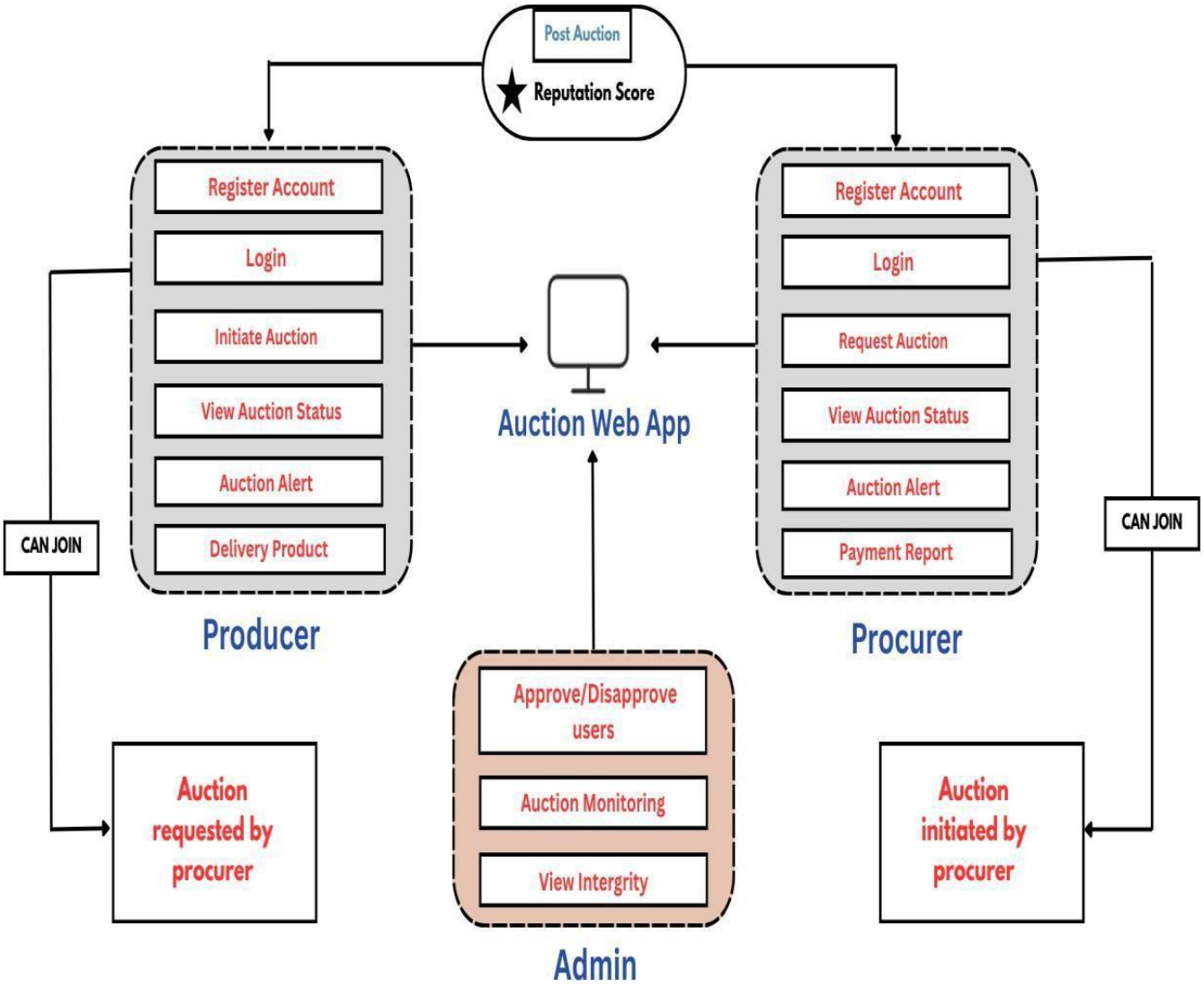


Fig 5.3 System Architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Table 5.3. Comparison of Existing Systems vs Proposed System

Feature	Traditional Market	Fixed-Price E-Commerce	Proposed System
Market Reach	Limited	Moderate	Wide (Online)
Intermediary Dependency	High	Medium	None
Bid Privacy	No	No	Yes (zk-SNARKs)
Identity Protection	No	No	Yes (Linked Signatures)
Real-Time Pricing	No	No	Yes
Trustability	No	No	Yes (Reputation System)
Data Tampering Protection	No	No	Yes (Blockchain)

5.4. MODULE LIST

5.4.1. Auction Simulator

The Auction Simulator forms the backbone of the entire platform, providing a virtual environment where producers and procurers can interact in real-time. Developed using Python Flask for backend logic, MySQL for data persistence, and Bootstrap for a responsive frontend, this module simulates the complete auction lifecycle—from creation to closure. It is designed to handle multiple concurrent auctions across diverse product categories, including agricultural goods, home appliances, livestock, and artworks.

When a producer initiates an auction, the simulator generates a unique auction entry with details such as product name, description, base price, auction type (sealed or open), and duration. Procurers browsing the platform can view active auctions, filtered by category or search terms, and participate by placing bids. The simulator continuously monitors auction timelines, automatically closing auctions when the deadline is reached and triggering the winner determination logic. All events—bid placements, auction closures, and winner announcements—are logged in real-time and displayed on user dashboards.

The simulator also integrates with the blockchain module to ensure that every critical action is recorded immutably. By simulating a decentralized marketplace, this module provides a realistic and secure environment for testing and real-world deployment. Its modular architecture allows for easy scaling and future enhancements, such as support for multi-currency transactions or AI-based price prediction.

5.4.2. End User

This module defines and manages the two primary user roles in the system: Producers (sellers) and Procurers (buyers). A secure registration and login system ensures that only authenticated users can access platform features. During registration, users provide essential details such as name, contact information, business type, and supporting documents (e.g., identity proof). All

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

user data is stored securely in the MySQL database, with passwords hashed using industry- standard algorithms.

- **Producers**

Producers are suppliers who list products for auction. Upon logging in, they access a personalized dashboard displaying their active auctions, past auction history, and pending payments. They can initiate new auctions by providing product details, images, and auction parameters. Producers can also participate in reverse auctions initiated by procurers, submitting sealed bids for procurement requests.

- **Procurers**

Procurers are buyers who join auctions to purchase products. Their dashboard shows active auctions they can join, auctions they have bid on, and any product requests they have initiated. Procurers can place bids in open auctions (with visible bid amounts) or submit sealed bids in reverse auctions. After winning an auction, they are guided through the payment process and can track order fulfillment.

- **Admin**

An additional administrative role oversees the platform. Admins approve or reject new user registrations based on submitted proofs, monitor auction activity for suspicious behavior, and can view the complete blockchain audit trail. The admin dashboard provides tools for user management, dispute resolution, and system monitoring.if needed.

5.4.3. Bidding Mechanism

This module handles the core functionality of placing bids, supporting both sealed and open bidding formats. It integrates advanced cryptographic algorithms to ensure privacy and fairness.

- **Sealed Bidding with zk-SNARKs:**

When a procurer initiates a reverse auction (or a producer initiates a forward auction requiring confidentiality), participants submit their bids in a sealed format. The system employs Zero-Knowledge Succinct Non- Interactive Arguments of Knowledge (zk-SNARKs) to cryptographically hide both the bidder's identity and the bid amount. Despite this secrecy, the auctioneer can mathematically verify that each bid is valid (e.g., within acceptable range) without accessing the actual bid value. This prevents any participant from adjusting their bid based on competitors' offers and eliminates bias in winner selection.

- **Open Bidding with Linked Ring Signatures:**

In open auction scenarios—typically when a producer lists an item for sale—bid amounts are visible to all participants to encourage competitive bidding and price discovery. However, to prevent collusion, bid shading, or post-auction retaliation, bidder identities are masked using Linked Ring Signatures. This algorithm allows a bidder to generate a signature that is indistinguishable from signatures produced by other members of a "ring" (group). While the system can verify that the bidder is an authorized participant, it cannot identify which specific member placed the bid. Crucially, the "linked" property allows detection if the same bidder places multiple bids, preventing ballot stuffing.

- **Commit-Reveal Scheme**

To further enhance fairness and prevent bid manipulation, a commit-reveal protocol is implemented. In the first phase (commit), bidders submit a commitment—a cryptographic hash of their bid amount concatenated with a randomly generated secret nonce. This commitment is stored on the blockchain, timestamping the bid without revealing its content. After the bidding phase closes, bidders enter the reveal phase, submitting their actual bid amount and nonce. The system recomputes the hash and verifies it against the stored commitment. This ensures that bids cannot be changed after submission, and no participant can gain an unfair advantage by seeing others' bids before revealing their own.

5.4.4 Reputation Scoring Module

Trust is a cornerstone of any marketplace. The Reputation Scoring Module incentivizes honest participation and builds

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

confidence among users by assigning a trust score to each producer and procurer based on their historical behavior. This module is critical for long-term platform sustainability, as it helps participants identify reliable trading partners and discourages fraudulent activities.

The reputation score is calculated using a weighted algorithm that considers multiple factors:

- **Successful Transactions**: Each completed auction where payment is made and the product is delivered (or received) as agreed adds positive points to the user's score.
- **Timely Participation**: Bidders who consistently place legitimate bids and honor their commitments receive incremental score increases.
- **Dispute Resolution:** Users involved in disputes that are resolved in their favor may regain points, while those found at fault lose points.
- **Feedback from Counterparties:** After each transaction, both parties can rate each other (e.g., on a 1–5 scale), contributing to the overall score.

The reputation score is displayed on user profiles and dashboards, allowing others to assess credibility before engaging in auctions. A higher score can lead to privileges such as priority listing in auction searches or reduced deposit requirements. Conversely, users with consistently low scores may be flagged for admin review or temporarily suspended. All reputation events—score updates, feedback submissions, and disputes—are recorded on the blockchain, ensuring an immutable and transparent history. This tamper-proof log prevents users from artificially inflating their scores and provides a reliable audit trail for the admin.

CHAPTER 6 - IMPLEMENTATION

6.1 MODEL BUILDING

The implementation of the Dual-Mode Blockchain Auction System involves developing secure, modular components for real-time bidding, blockchain integration, user interaction, and cryptographic privacy. The system is built using Python Flask for the backend, MySQL for relational data storage, and a custom private blockchain for immutable audit trails. Advanced cryptographic algorithms—zk-SNARKs (simulated via Schnorr proofs on elliptic curves) for sealed bidding and Linked Ring Signatures (RSA-based) for open bidding—are integrated to ensure bidder anonymity and bid confidentiality. A Commit-Reveal Scheme prevents bid manipulation, while a Reputation Score Algorithm encourages honest participation. The web interface uses HTML5, CSS3, Bootstrap 5, and JavaScript, providing role-specific dashboards for producers, procurers, and administrators.

6.1.1 Database Design

The MySQL database double_auction_dual contains the following core tables:

- **da_producer –** stores producer details (ID, password, company name, full name, mobile, proof, email, reputation_score, public_key, approved_status).
- **da_procurer** – similar structure for procurers.
- **da_admin** – stores admin credentials.
- **auctions** – records each auction (initiator_type, initiator_id, product_name, description, base_price, auction_type, start_time, end_time, status, winner_id, final_price, payment_status, delivery_status).
- **bids** – stores all bids (auction_id, bidder_type, bidder_id, bid_amount, bid_hash, revealed, timestamp). The bid_hash field holds cryptographic data (zero-knowledge proof or ring signature) as a JSON string.
- **notifications** – manages in-app notifications (user_type, user_id, message, link, created_at, is_read).

Foreign key relationships maintain data consistency, and all critical actions are logged into the blockchain.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
CREATE TABLE auctions (

    auction_id INT AUTO_INCREMENT PRIMARY KEY,

    initiator_type ENUM('producer','procurer') NOT NULL, initiator_id
    VARCHAR(50) NOT NULL,  product_name VARCHAR(255) NOT
    NULL, description TEXT,
    base_price DECIMAL(10,2) NOT NULL, auction_type
    ENUM('open','sealed') NOT NULL, start_time DATETIME NOT
    NULL,
    end_time DATETIME NOT NULL,

    status ENUM('active','closed') DEFAULT 'active', winner_id
    VARCHAR(50),
    final_price DECIMAL(10,2),

    payment_status ENUM('pending','paid') DEFAULT 'pending', delivery_status
    ENUM('pending','delivered') DEFAULT 'pending', INDEX (initiator_id),
    INDEX (status)

);
```

6.1.2 Blockchain Implementation

A custom Blockchain class manages the private ledger. Blocks contain an index, timestamp, list of transactions, proof (hash of previous block + timestamp), and previous hash. Each block's hash is computed using SHA-256. The chain is stored in dualchain.json and loaded on startup. Whenever a critical event occurs (user registration, auction creation, bid placement, auction closure, payment), a new transaction is added and a new block is mined, ensuring tamper-proof records.

```
class Blockchain:
    def ____init____(self):
        self.chain = [] self.current_transactions =
        [] self.load_chain()
    def new_block(self, proof, previous_hash=None): block = {
            'index': len(self.chain) + 1,

            'timestamp': str(datetime.datetime.now()),
            'transactions': self.current_transactions, 'proof':
            proof,
            'previous_hash': previous_hash or self.hash(self.chain[-1])

        }

        block['hash'] = self.hash(block)
        self.current_transactions = []
        self.chain.append(block) self.save_chain()
        return
    block
    @staticmetho
    d               def
    hash(block)


        block_string = json.dumps(block, sort_keys=True).encode() return
        hashlib.sha256(block_string).hexdigest()
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.1.3 Cryptographic Primitives

- **Sealed Bidding with zk-SNARKs**

To hide both bidder identity and bid value, we generate a zero-knowledge proof that the bidder knows the bid amount without revealing it. The proof is stored in bid_hash along with a public key. During the reveal phase, the bidder submits the actual amount, and the system verifies the proof.

```
def generate_zk_proof(secret_value, nonce):

    secret_int = int.from_bytes(hashlib.sha256(str(secret_value).encode()).digest(), 'big') %
SECP256k1.order

    r = int(nonce, 16) % SECP256k1.order g =
    SECP256k1.generator commitment_point = g
    * r public_key_point = g * secret_int
    challenge_data = commitment_point.to_bytes() +
public_key_point.to_bytes()

    challenge = int.from_bytes(hashlib.sha256(challenge_data).digest(), 'big') %

SECP256k1.order

    response = (r + challenge * secret_int) % SECP256k1.order proof = {

'commitment': base64.b64encode(commitment_point.to_bytes()).decode(), 'response': hex(response)[2:]
    }

    return proof, public_key_point.to_bytes().hex()
```

- **Open Bidding with Linked Ring Signatures**

In open auctions, bid amounts are visible but bidder identities are masked using a ring signature. The signature is created using the bidder's private key and a ring of public keys from all potential bidders of the opposite role. Verification ensures the signature came from a member of the ring without revealing which one.

```
def ring_sign(ring_public_keys, signer_private_key, message): h =
    SHA256.new(message)
    signature = pkcs1_15.new(signer_private_key).sign(h) ring_pem =
    [key.publickey().export_key().decode() for key in
ring_public_keys]

    return {'ring': ring_pem, 'signature': base64.b64encode(signature).decode()}
```

- **Commit-Reveal Scheme**

For sealed auctions, the system implements a commit-reveal protocol. Upon bid placement, the bidder submits a commitment (hash of bid + nonce). Later, they reveal the actual bid and nonce, which are verified against the stored commitment. This prevents bid changes after seeing others' bids.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.1.4 User Authentication and Role Management

Three types of users are supported: producers, procurers, and admin. Passwords are hashed using SHA-256. After registration, admin approval is required. Sessions track the logged-in user and their role. Notifications inform users of important events.

```
@app.route('/login', methods=['GET', 'POST']) def login():
    if request.method == 'POST':

        # Validate producer credentials

        hashed_pwd = hashlib.sha256(request.form['pass'].encode()).hexdigest() cursor.execute('SELECT * FROM
        da_producer WHERE producer_id = %s
AND password = %s AND approved_status=1', (uname,
                    hashed_pwd))
        if account:

            session['username'] = uname
            session['role'] = 'producer'
            return redirect(url_for('producer_dashboard'))

    # ...
```

6.1.5 Auction Lifecycle

Auction Creation – Producers or procurers can create an auction by providing product details, base price, auction type (sealed/open), and end time. The auction is stored in the auctions table with status active.

Placing a Bid – Depending on the auction type, the appropriate cryptographic method is used. For sealed bids, a zero-knowledge proof is generated; for open bids, a ring signature is created. The bid record stores the cryptographic data in bid_hash. A notification is sent to the auction initiator.

```
@app.route('/place_bid/<int:auction_id>',          methods=['POST'])          def
place_bid(auction_id):
    # ... determine auction type

    if auction['auction_type'] == 'sealed':
proof, public_key = generate_zk_proof(bid_amount, nonce) combined = json.dumps({'proof': proof,
'public_key': public_key})
            cursor.execute('INSERT INTO bids ... (bid_amount=0, bid_hash=%s, revealed=FALSE)',
        (combined,))
    else:  # open

        ring_sig = ring_sign(ring_keys, signer_key, message) combined =
        json.dumps(ring_sig)
        cursor.execute('INSERT INTO bids ... (bid_amount=%s, bid_hash=%s)', (bid_amount,
    combined))
```

Closing an Auction – Admin manually closes an auction after its end time. For sealed auctions, only revealed bids are considered. Winner is determined as the highest bid (if producer-initiated) or lowest bid (if procurer-initiated). The winner receives +10 reputation.

Payment and Delivery – A dummy payment route marks the auction as paid and delivered, awarding another +10 reputation to the winner.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

6.1.6 Reputation System

The reputation score for each user is stored in the reputation_score column of the producer/procurer tables. It is updated after winning an auction and after completing payment. Higher reputation increases trust and visibility.

```
def add_reputation(user_type, user_id, points):
    table = 'da_producer' if user_type == 'producer' else 'da_procurer' cursor.execute(f'UPDATE {table}
    SET reputation_score = reputation_score +
%s WHERE {user_type}_id = %s',
                (points, user_id))
```

6.1.7 Notifications

In-app notifications are stored in the notifications table. They are displayed on user dashboards and can be marked as read. Notifications are triggered for logins, approvals, new bids, auction closure, and payment completion.

6.1.8 Admin Dashboard

The admin dashboard displays pending user approvals, auctions that have ended and need closing, and system notifications. Admin can approve users and close auctions via simple button actions.

6.1.9 Blockchain Viewer

Admin can view the entire blockchain at /view_block, where each block's transactions and hash are displayed, ensuring full transparency.

This implementation realizes a fully functional dual-mode auction platform that guarantees privacy, fairness, transparency, and trust through the integration of blockchain and advanced cryptography. All core modules—user management, auction creation, bidding, winner selection, reputation, and immutable logging— are operational and tested.

CHAPTER 7 - TESTING

Testing is a critical phase to ensure the reliability, security, and usability of the Dual-Mode Blockchain Auction System. The following testing strategies were employed.

7.1 Unit Testing

Unit tests were conducted on individual modules such as user registration, bid submission, winner determination, and blockchain logging. Each function was tested with both valid and invalid inputs to verify correct behaviour.

For example, the winner determination logic was tested with various bid sets to ensure that the highest bidder (in open auctions) or the lowest bidder (in procurement auctions) is correctly selected.

7.2 Integration Testing

Integration testing verified that different modules interact with each other correctly. The key integration points tested include:

- Interaction between the bidding module and the blockchain logging module.
- Data flow between the user interface and the database.
- Communication between the frontend (Flask) and the MySQL database.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

7.3 System Testing

System testing evaluated the entire system as a complete unit in a simulated environment with multiple concurrent users (producers and procurers).
The following scenarios were tested:

- Simultaneous auction creation and bid submission.
- Sealed-bid and open-bid auctions running concurrently.
- Payment processing and reputation updates after auction completion.
- The system successfully handled these scenarios without functional errors.

7.4 Performance Testing

Performance testing was conducted to measure system responsiveness under load conditions. Using Apache JMeter, the system simulated up to 100 concurrent users placing bids simultaneously.

Key performance metrics observed were:

Metric	Result
Average response time for bid submission	< 2 seconds
Blockchain block creation latency	< 5 seconds
Database query execution time	< 0.5 seconds

These results indicate that the system performs efficiently even under moderate load.

7.5 Security Testing

Security testing was conducted to identify potential vulnerabilities and ensure data protection.

- **Authentication & Authorization**: Tested against common threats such as SQL injection, session hijacking, and privilege escalation.
- **Cryptographic Implementation**: Verified that zk-SNARKs and Linked Ring Signatures correctly preserve bidder anonymity and protect sensitive bid data.
- **Blockchain Integrity**: Attempts were made to tamper with recorded auction data. Any modification resulted in hash mismatches, proving the immutability of the blockchain.

7.6 Usability Testing

Usability testing involved 10 participants (5 producers and 5 procurers).

Each participant was asked to perform common tasks including:

- User registration
- Creating an auction
- Placing bids
- Viewing auction results

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Feedback was collected through a questionnaire after task completion. The system achieved a 90% user satisfaction rate, with participants highlighting the intuitive interface, clear instructions, and smooth bidding workflow.

CHAPTER 8 - RESULTS AND DISCUSSION

8.1 OUTPUT

Fig 8.1.1 Dual-Blockbid Home Page

In Fig 8.1.1, shows the home page of the Dual-BlockBid – which is dual mode blockchain based auction platform, where producers and procurers can initiate and join in auction respectively. There is separate login page for producer,procurer and admin which navigates them to their desired dashboard.

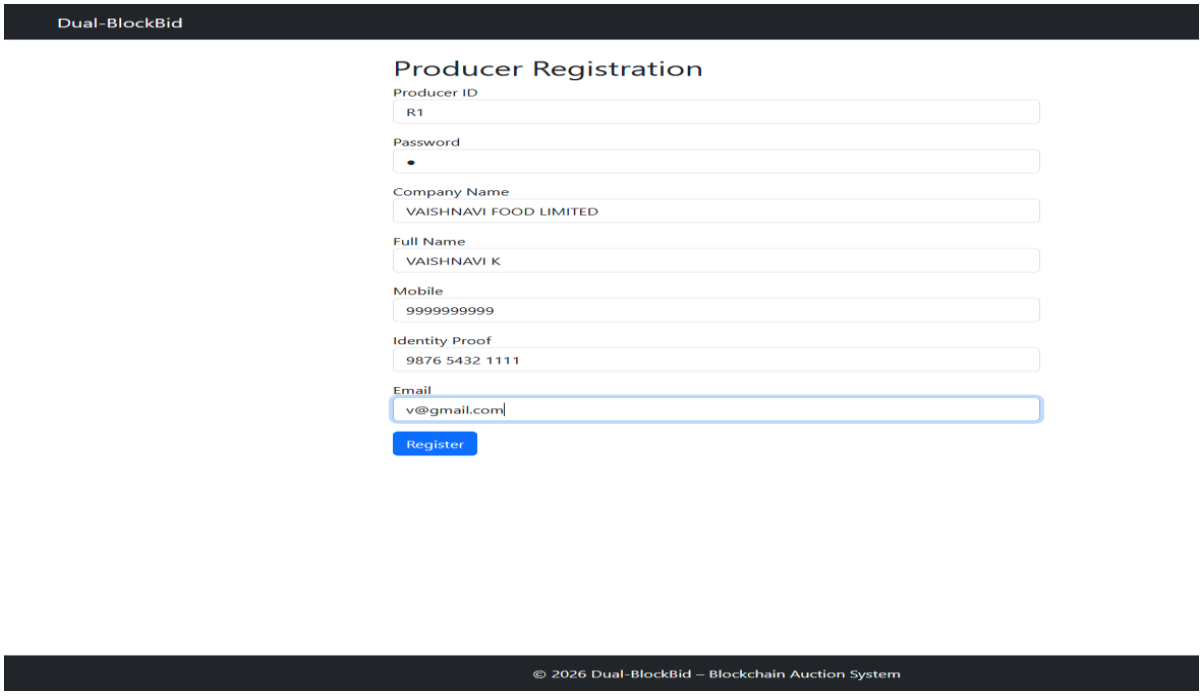


Fig 8.1.2 Producer Registration Page

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

This is the registration page of the producer, in which the producer is asked to enter their details such as company name (if personal, NA), full name, mobile no., any proof, email and with password as shown in Fig 8.1.2. And like this, there is a separate registration page for procurer side.

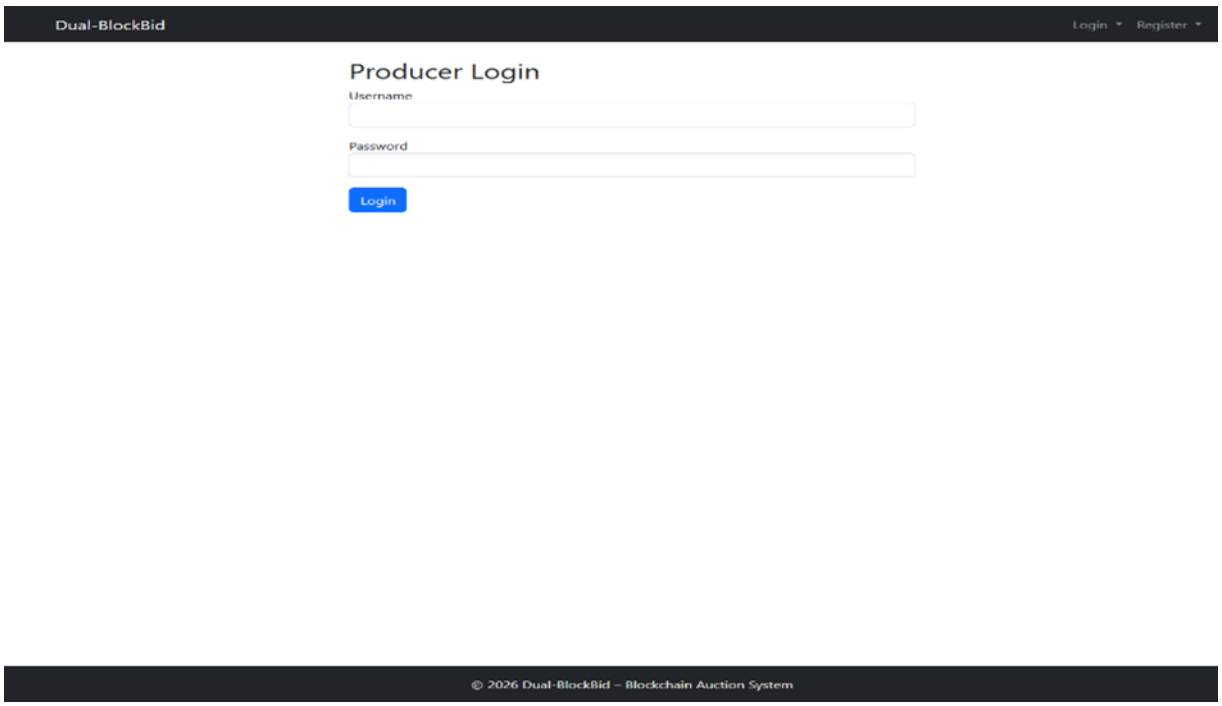


Fig 8.1.3 Producer Login Page

In Fig 8.1.3 shows the login page of the producer where producer can login into their dashboard using their registered username and password.

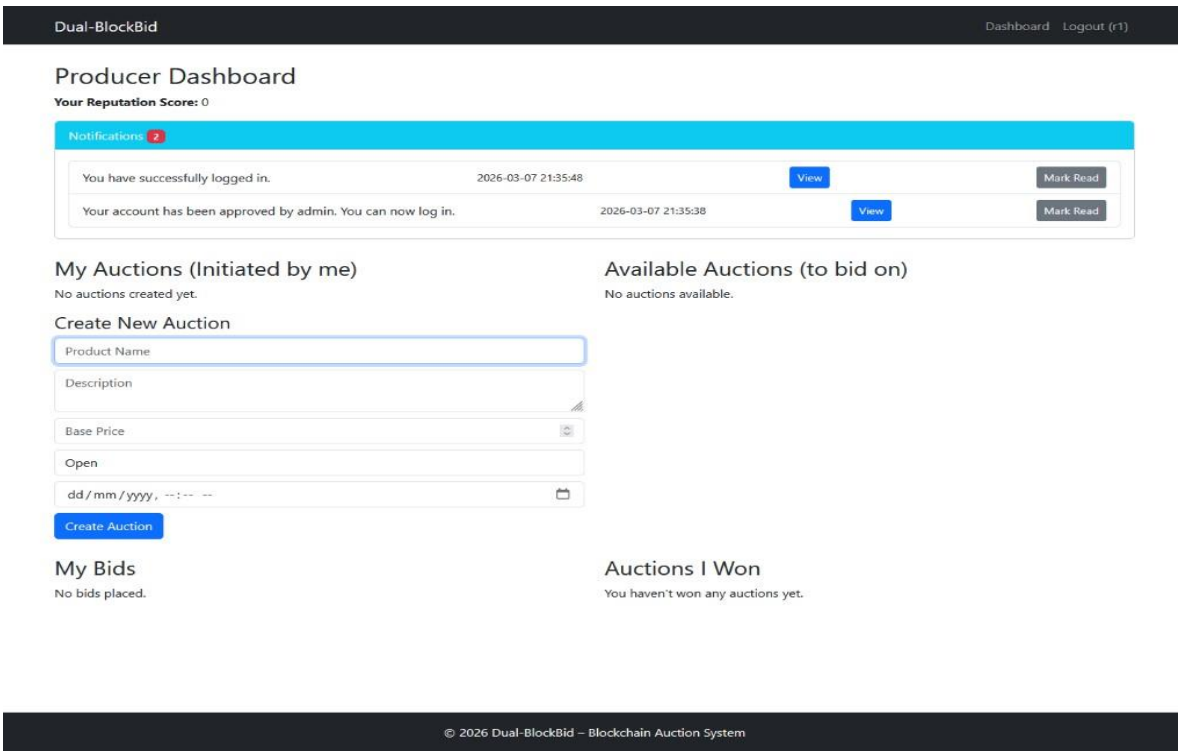


Fig 8.1.4 Producer Dashboard

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

In Fig 8.1.4 represents the dashboard of the producer - where the details of the producers are displayed in cards, the auction initiated by producers are shown in their dashboard to see the auction status i.e., who joins in auction and what bid amounts are being done.

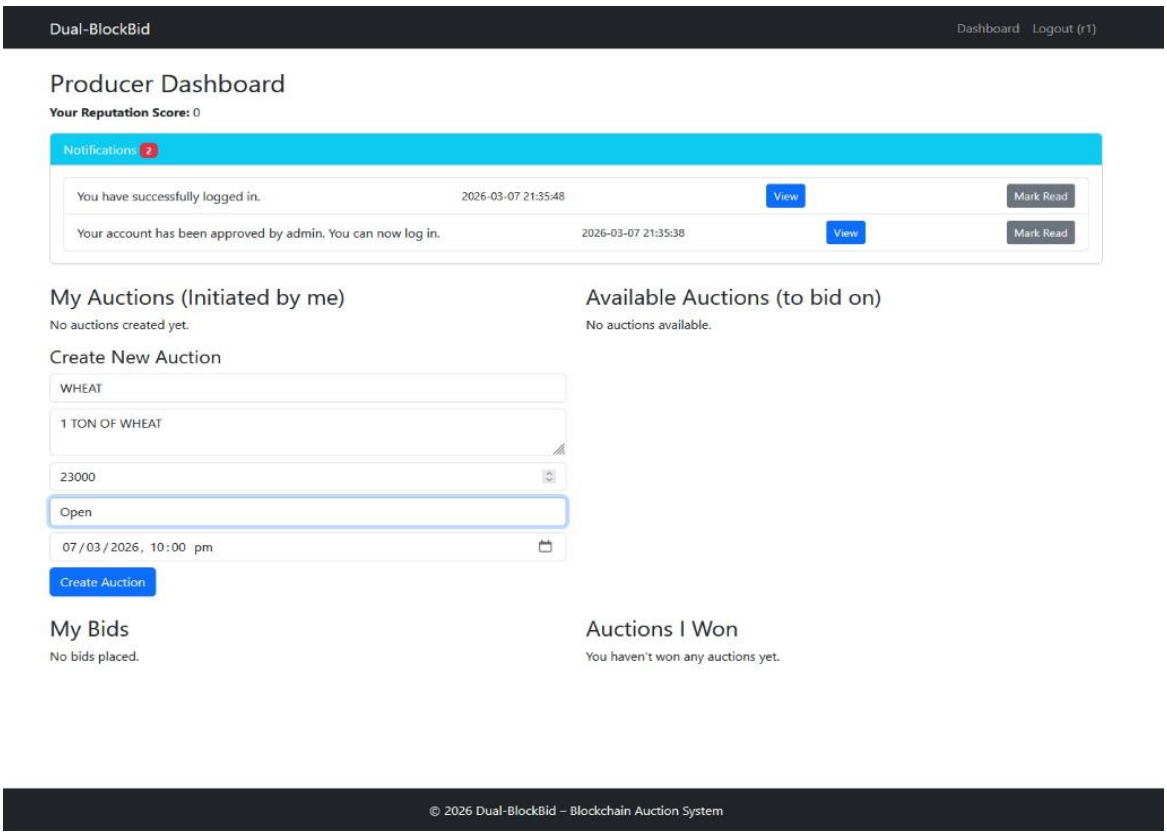


Fig 8.1.5 Initiate / Request Auction

In Fig 8.1.5 shows the auction creation frame, where the producers initiate an auction by adding the details of the desired product and on the other hand the procurers can request an auction by naming the deatils of the particular product.

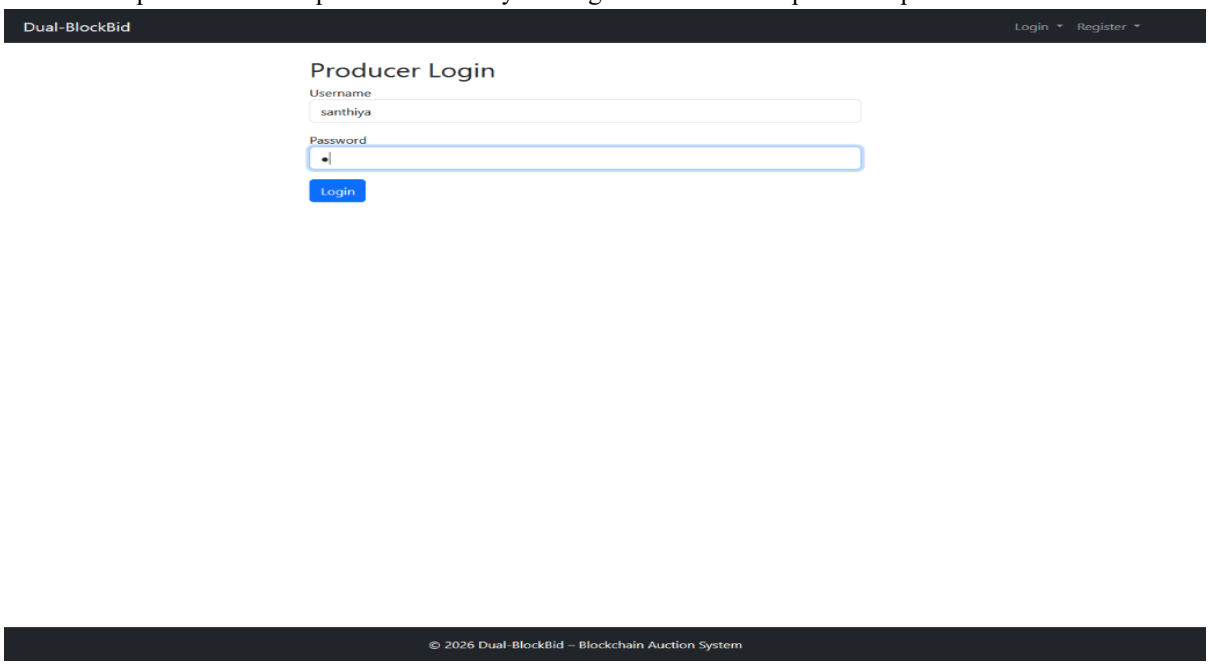


Fig 8.1.6 Procurer Login Page

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

This is the login page of the procurer, where one can login into their dashboard using their registered username and password as shown in Fig 8.1.6

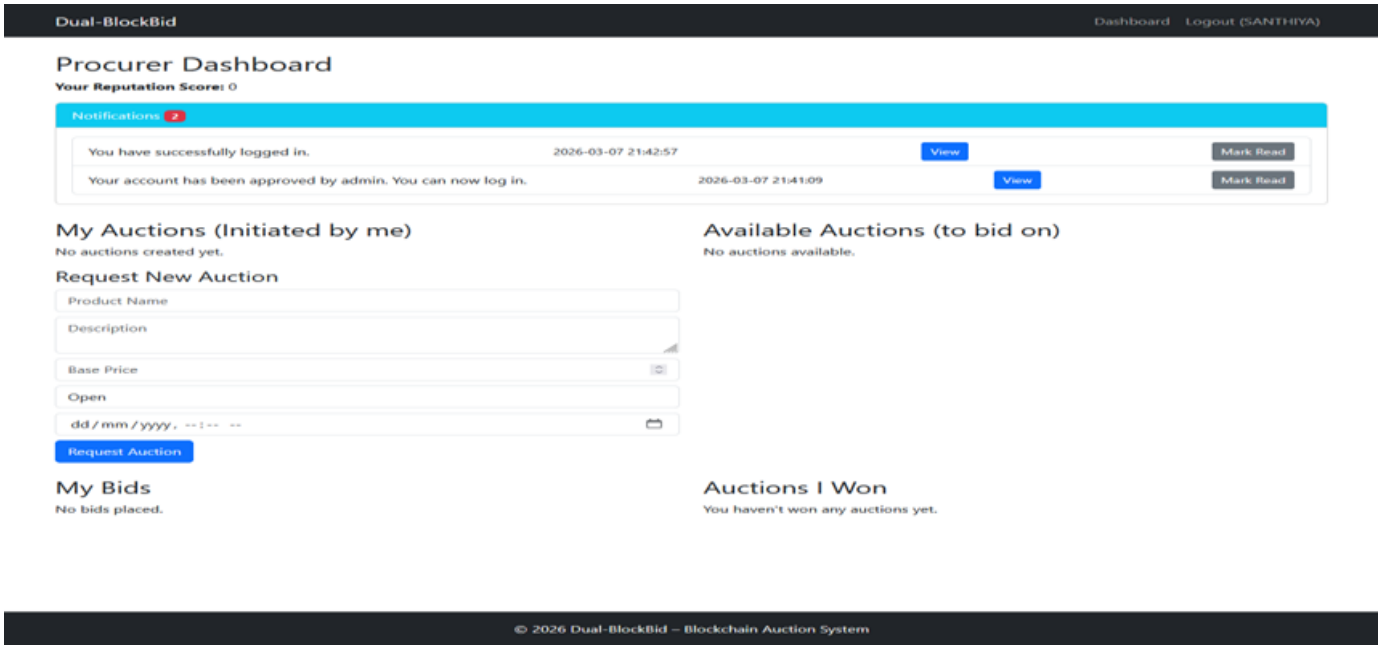


Fig 8.1.7 Procurer Dashboard

The Dashboard of the procurer is shown in Fig 8.1.7, where particular user can view their profile and the auctions created by producers, in that one can join by bidding amount. Also it contains separate frame for product request – where one can request an auction and view reports to pay amount if wins and producer will delivery product respectively.

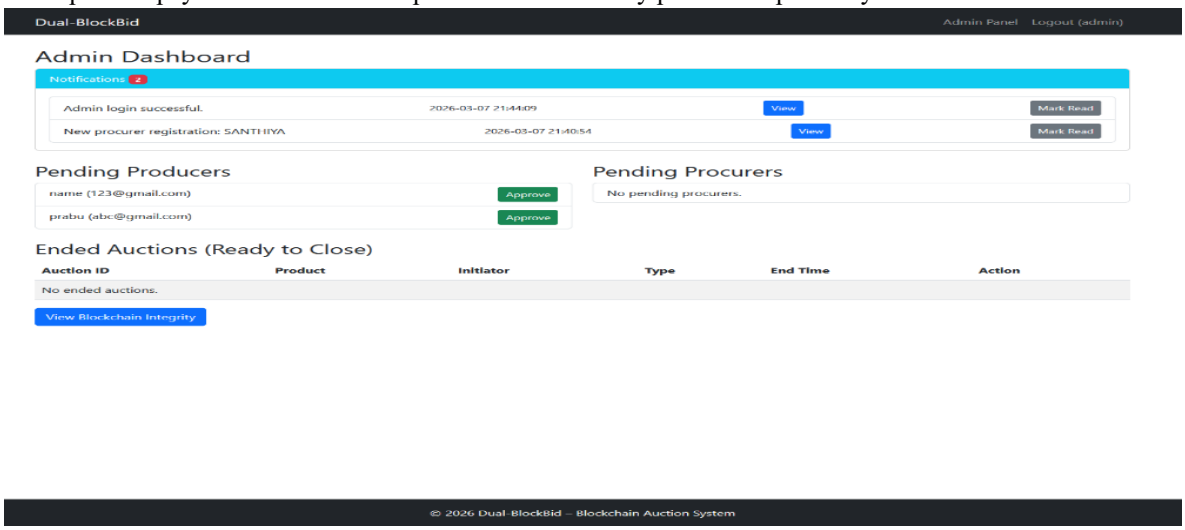


Fig 8.1.8 Admin Dashboard

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The Fig 8.1.8 shows the admin dashboard, in which admin will approve and disapprove the producers and procurers based on proof and other required details.

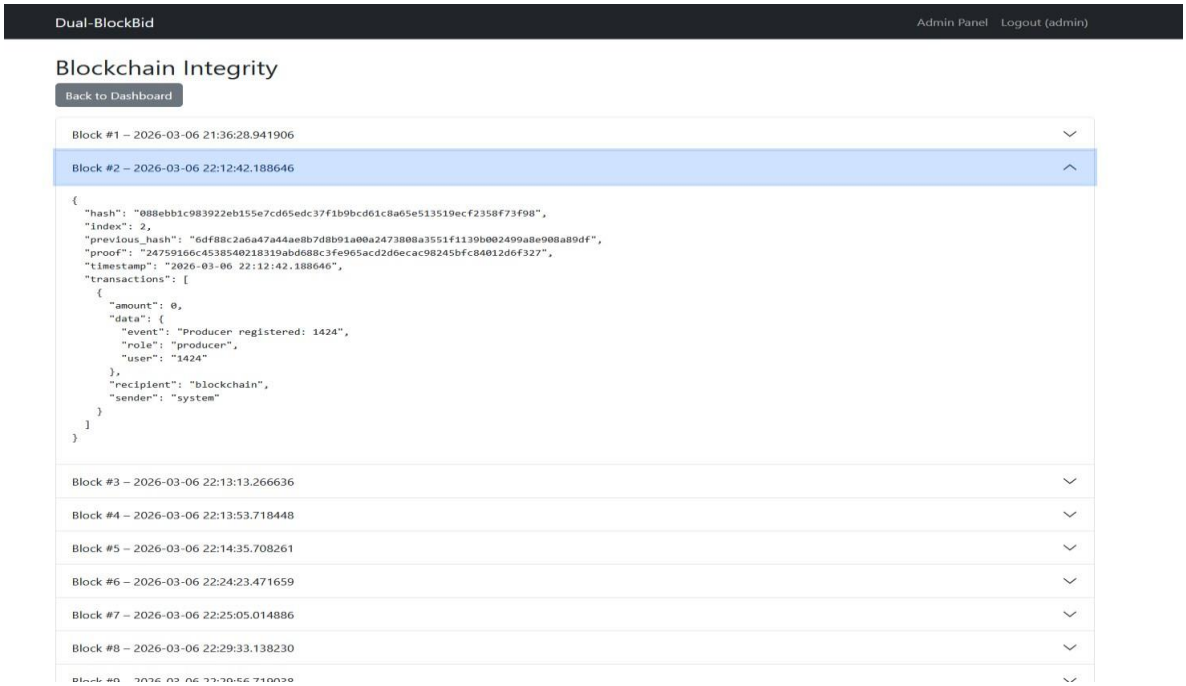


Fig 8.1.9 Integrity Viewpoint

In Fig 8.1.9 shows the view integrity in admin dashboard – where the private blockchain has been integrated for securely storing the auction details and winner data, when admin clicks the traceability key the original stored data can be viewed.

CHAPTER 9 - CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

The proposed Blockchain-Based Dual-Mode Auction System offers a transformative solution for secure, transparent, and privacy-preserving digital marketplaces across diverse domains, including agricultural produce, home appliances, artworks, and more as multi category sectors. By integrating techniques such as zk-SNARKs for sealed bidding to hide the bidders' anonymity and bid values, Linked Ring Signatures for open bidding to hide identity and reveal bid values for fair competition and auction and Commit- Reveal Scheme. Additionally a Reputation Score is implemented for trust score which relies on bidder's participation, auction winning that make way for future auction participation. Furthermore, a private blockchain system, where the system ensures confidential bidding, fair winner determination, and records auction data securely. It removes the need for intermediaries, supports both sealed and open bidding modes, and fosters direct, trust-based engagement between buyers and sellers. With features like real-time auction capabilities, immutable auction logs, and a reputation-driven trading environment, the platform enhances market efficiency and participant accountability. Its modular, scalable architecture makes it a robust, future-ready foundation for building trustworthy and decentralized digital marketplaces across multiple sectors.

9.2. FUTURE ENHANCEMENT

- **AI-Powered Price Prediction and Analytics:** Integrating artificial intelligence (AI) and machine learning (ML) algorithms to predict market trends and price fluctuations based on historical data and real-time market conditions. This can help both producers and procurers make informed bidding decisions and optimize auction strategies.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- **Mobile Application Integration:** Developing a mobile application for both producers and procurers to provide on-the-go access to auctions, enabling them to participate in real-time from any location. This can improve user engagement and increase auction participation, especially for small-scale farmers.

- **Smart Contract Enhancements:** Expanding the functionality of smart contracts to automatically handle more complex tasks, such as post- auction logistics, including shipment tracking and quality checks, ensuring that transactions are not only completed but also fulfilled to the satisfaction of all parties involved.

- **Multi-Currency and Cross-Border Transactions:** Adding support for multiple cryptocurrencies and fiat currencies to facilitate international trade. This would enable farmers and traders from different countries to participate, broadening the scope of the auction system to global markets.

- **AI-Based Bid Pattern Analysis Integrate AI/ML** to Detect abnormal bidding behavior Identify collusion between bidders Predict fair market value of products Recommend optimal base price to producers.

- **Multi-Currency & Crypto Payment Support Enhance payment module to support**: Cryptocurrency payments Stablecoins Automatic currency conversion Cross-border auctions.

- **Real-Time Live Auction Streaming** Add Live streaming feature for premium auctions Real-time countdown timers Live bid animation dashboard Auction host moderation

APPENDIX – I CODING

app.py - Flask file

```
import os
import json
import
hashlib
import
datetime
import
random
import
base64
from flask import Flask, render_template, request, session, redirect, url_for, flash
import mysql.connector
from mysql.connector import Error from
Crypto.PublicKey import RSA from
Crypto.Signature import pkcs1_15 from
Crypto.Hash import SHA256
from ecdsa import SigningKey, VerifyingKey, SECP256k1 app = Flask(
__name__)
app.secret_key = 'your-secret-key-here' def
get_db_connection():
    try:
        conn = mysql.connector.connect(
            host="127.0.0.1",
            user="root", password="@Bcpl1234",
            database="double_auction_dual"
        )
        return conn
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
except Error as
e:
print(f" Database connection error: {e}") return None
def add_notification(user_type, user_id, message, link='#'): conn =
get_db_connection()
if not
conn:
return
cursor = conn.cursor()
try:
cursor.execute('''
INSERT INTO notifications (user_type, user_id, message, link, created_at, is_read)
VALUES (%s, %s, %s, %s, NOW(), FALSE)
''', (user_type, user_id, message, link)) conn.commit()
except Error as e: print(f"Notification
error: {e}")
finally:
cursor.close(
)
conn.close()
def get_unread_notifications(user_type, user_id): conn =
get_db_connection()
if not
conn:
return
[]
cursor = conn.cursor(dictionary=True)
cursor.execute('''
SELECT * FROM notifications
WHERE user_type = %s AND user_id = %s AND is_read = FALSE ORDER BY created_at
DESC
''', (user_type, user_id)) notifications =
cursor.fetchall() cursor.close()
conn.close()
return notifications
def mark_notification_read(notification_id): conn =
get_db_connection()
if not
conn:
return
cursor = conn.cursor()
cursor.execute('UPDATE notifications SET is_read = TRUE WHERE id =
%s', (notification_id,))
conn.commit()
cursor.close()
conn.close()
class Blockchain:
def ____init____(self):
self.chain = [] self.current_transactions = [] self.load_chain()
def load_chain(self):
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                try:
                    if os.path.exists('dualchain.json'): with
                        open('dualchain.json', 'r') as f:
                            data = json.load(f)
                            self.chain = data if isinstance(data, list) else [] if not
                    self.chain:
                        self.new_block(previous_hash='1', proof=100) except
                Exception as e:
                    print(f"Blockchain load error: {e}")
                    self.new_block(previous_hash='1', proof=100)
            def new_block(self, proof, previous_hash=None): block = {
                    'index': len(self.chain) + 1,
                    'timestamp': str(datetime.datetime.now()),
                    'transactions': self.current_transactions, 'proof':
                    proof,
                    'previous_hash': previous_hash or (self.hash(self.chain[-1]) if self.chain else '1')
                }
                block['hash'] = self.hash(block)
                self.current_transactions = []
                self.chain.append(block) self.save_chain()
                return block
            def new_transaction(self, sender, recipient, amount, data): self.current_transactions.append({
                    'sender': sender,
                    'recipient': recipient,
                    'amount': amount,
                    'data': data
                })
                return self.last_block['index'] + 1 if self.chain else 1
            @staticmethod
def hash(block):
                block_string = json.dumps(block, sort_keys=True).encode() return
                hashlib.sha256(block_string).hexdigest()
            @property
            def last_block(self):
                return self.chain[-1] if self.chain else None def
            save_chain(self):
                with open('dualchain.json', 'w') as f: json.dump(self.chain, f,
                    indent=4)
        blockchain = Blockchain()
        def add_block_to_chain(data, user_type, user_id):
            blockchain.new_transaction(
                sender='system',
                recipient='blockchain', amount=0,
                data={'event': data, 'user': user_id, 'role': user_type}
            )
            last_hash = blockchain.hash(blockchain.last_block) if blockchain.last_block else '1'
            proof = hashlib.sha256((last_hash + str(datetime.datetime.now())).encode()).hexdigest()
            blockchain.new_block(proof=proof, previous_hash=last_hash) def
        generate_commitment(bid_value, nonce):
            return hashlib.sha256(f"{bid_value}{nonce}".encode()).hexdigest()
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
def verify_commitment(bid_value, nonce, commitment):
    return generate_commitment(bid_value, nonce) == commitment def
generate_zk_proof(secret_value, nonce):
     secret_int = int.from_bytes(hashlib.sha256(str(secret_value).encode()).digest(), 'big') %
SECP256k1.order
     r = int(nonce, 16) % SECP256k1.order g =
    SECP256k1.generator commitment_point = g
    * r public_key_point = g * secret_int
    challenge_data = commitment_point.to_bytes() +
public_key_point.to_bytes()
challenge = int.from_bytes(hashlib.sha256(challenge_data).digest(), 'big') %
SECP256k1.order
    response = (r + challenge * secret_int) % SECP256k1.order proof = {
        'commitment': base64.b64encode(commitment_point.to_bytes()).decode('utf-
8'),
        'response': hex(response)[2:]
    }
    return proof, public_key_point.to_bytes().hex()
def verify_zk_proof(proof, public_key_hex, challenge_data=None): curve =
    SECP256k1
    g = curve.generator
    public_key_bytes = bytes.fromhex(public_key_hex)
    public_key = VerifyingKey.from_string(public_key_bytes, curve=curve) commitment_bytes =
    base64.b64decode(proof['commitment'])
    commitment = VerifyingKey.from_string(commitment_bytes, curve=curve) response =
    int(proof['response'], 16) % curve.order
    if challenge_data is None:
        challenge_data = commitment_bytes + public_key_bytes
    challenge = int.from_bytes(hashlib.sha256(challenge_data).digest(), 'big') % curve.order
    lhs = g * response
    rhs = commitment.pubkey.point + public_key.pubkey.point * challenge return lhs == rhs
def generate_rsa_keypair(): key =
    RSA.generate(2048) return
    key
def ring_sign(ring_public_keys, signer_private_key, message):

    h = SHA256.new(message)
    signature = pkcs1_15.new(signer_private_key).sign(h) ring_pem =
    []
    for key in ring_public_keys:
        if isinstance(key, RSA.RsaKey):
            ring_pem.append(key.publickey().export_key().decode('utf-8'))
        else:
            ring_pem.append(key)
return {
        'ring': ring_pem,
        'signature':  base64.b64encode(signature).decode('utf-8')
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    }
def verify_ring_signature(ring_signature, message):
        ring = [RSA.import_key(key) for key in ring_signature['ring']] signature =
    base64.b64decode(ring_signature['signature'])
    h = SHA256.new(message) for
    pub_key in ring:
        try:
            pkcs1_15.new(pub_key).verify(h, signature) return True
        except (ValueError, TypeError): continue
    return False
def add_reputation(user_type, user_id, points): conn =
    get_db_connection()
    if not
        conn:
        return
    cursor = conn.cursor()
    table = 'da_producer' if user_type == 'producer' else 'da_procurer' cursor.execute(f'UPDATE {table} SET
    reputation_score = reputation_score +
%s WHERE {user_type}_id = %s',
                    (points,
    user_id)) conn.commit()
    cursor.close()   conn.close()
def get_reputation(user_type, user_id): conn =
    get_db_connection()
    if not
        conn:
        return
        0
    cursor = conn.cursor(dictionary=True)
    table = 'da_producer' if user_type == 'producer' else 'da_procurer' cursor.execute(f'SELECT reputation_score
    FROM {table} WHERE
{user_type}_id = %s', (user_id,)) result =
    cursor.fetchone() cursor.close()

    conn.close()
    return result['reputation_score'] if result else 0 @app.route('/')
def index():
    return render_template('web/index.html')
@app.route('/login', methods=['GET', 'POST']) def login():
    if request.method == 'POST':
        conn = get_db_connection() if
        not conn:
            return "Database unavailable", 500 cursor =
        conn.cursor(dictionary=True) uname =
        request.form['uname']
        pwd = request.form['pass']
        hashed_pwd = hashlib.sha256(pwd.encode()).hexdigest() cursor.execute('SELECT * FROM
        da_producer WHERE producer_id = %s
AND password = %s AND approved_status=1', (uname,
                        hashed_pwd))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                account = cursor.fetchone() cursor.close()
                conn.clos
                e() if
                account:
                    session['username'] = uname
                    session['role'] = 'producer'
                    add_notification('producer', uname, 'You have successfully logged in.', url_for('producer_dashboard'))
                    return redirect(url_for('producer_dashboard')) else:
                    return render_template('web/login.html', msg='Invalid credentials or not approved')
            return render_template('web/login.html')
        @app.route('/login_prc', methods=['GET', 'POST']) def
        login_prc():
            if request.method == 'POST':
                conn = get_db_connection() if
                not conn:
                    return "Database unavailable", 500
cursor = conn.cursor(dictionary=True) uname = request.form['uname']
                pwd = request.form['pass']
                hashed_pwd = hashlib.sha256(pwd.encode()).hexdigest() cursor.execute('SELECT * FROM
                da_procurer WHERE procurer_id = %s
        AND password = %s AND approved_status=1', (uname,
                                hashed_pwd))
                account = cursor.fetchone() cursor.close()
                conn.clos
                e() if
                account:
                    session['username'] = uname
                    session['role'] = 'procurer'
                    add_notification('procurer', uname, 'You have successfully logged in.', url_for('procurer_dashboard'))
                    return redirect(url_for('procurer_dashboard')) else:
                    return render_template('web/login_prc.html', msg='Invalid credentials or not approved')
            return render_template('web/login_prc.html')
        @app.route('/login_admin', methods=['GET', 'POST']) def
        login_admin():
            if request.method == 'POST':
                conn = get_db_connection() if
                not conn:
                    return "Database unavailable", 500 cursor =
                conn.cursor(dictionary=True)
                cursor.execute('SELECT * FROM da_admin WHERE username = %s AND password = %s',
                                (request.form['uname'], request.form['pass'])) account =
                cursor.fetchone()
                cursor.clo
                se()
                conn.clos
                e() if
                account:
                    session['username'] = request.form['uname'] session['role'] =
                    'admin'
                    add_notification('admin', session['username'], 'Admin login successful.',
url_for('admin_dashboard'))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
            return redirect(url_for('admin_dashboard')) else:
            return render_template('web/login_admin.html', msg='Invalid credentials')
    return render_template('web/login_admin.html')
@app.route('/register_producer', methods=['GET', 'POST']) def
register_producer():
    if request.method == 'POST':
        password = hashlib.sha256(request.form['password'].encode()).hexdigest() conn =
        get_db_connection()
        if not conn:
            return "Database unavailable", 500 cursor =
        conn.cursor()
        key = generate_rsa_keypair()
        public_key_pem = key.publickey().export_key().decode('utf-8') cursor.execute('''
            INSERT INTO da_producer (producer_id, password, company_name, full_name, mobile,
proof, email, reputation_score, public_key)
            VALUES (%s, %s, %s, %s, %s, %s, %s, 0, %s)
        ''', (request.form['producer_id'], password, request.form['company'], request.form['fullname'],
              request.form['mobile'], request.form['proof'],
request.form['email'], public_key_pem))
        conn.commit()
        cursor.close(
        )
        conn.close()
        add_block_to_chain(f"Producer registered: {request.form['producer_id']}", "producer",
request.form['producer_id'])
        add_notification('admin', 'admin', f"New producer registration:
{request.form['producer_id']}", url_for('admin_dashboard')) return
        redirect(url_for('login'))
    return render_template('web/register_producer.html')
@app.route('/register_procurer', methods=['GET', 'POST']) def
register_procurer():
    if request.method == 'POST':
        password = hashlib.sha256(request.form['password'].encode()).hexdigest() conn =
        get_db_connection()


        if not conn:
            return "Database unavailable", 500 cursor =
        conn.cursor()
        key = generate_rsa_keypair()
        public_key_pem = key.publickey().export_key().decode('utf-8') cursor.execute('''
            INSERT INTO da_procurer (procurer_id, password, full_name, mobile, proof, email,
reputation_score, public_key)
            VALUES (%s, %s, %s, %s, %s, %s, 0, %s)
        ''', (request.form['procurer_id'], password, request.form['fullname'], request.form['mobile'],
              request.form['proof'], request.form['email'],
public_key_pem))
        conn.commit
        ()
        cursor.close(
        )
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
            conn.close()
            add_block_to_chain(f"Procurer registered: {request.form['procurer_id']}", "procurer",
        request.form['procurer_id'])
            add_notification('admin', 'admin', f"New procurer registration:
        {request.form['procurer_id']}", url_for('admin_dashboard')) return
            redirect(url_for('login_prc'))
        return render_template('web/register_procurer.html') @app.route('/producer_dashboard')
        def producer_dashboard():
        if 'username' not in session or session['role'] != 'producer': return
            redirect(url_for('login'))
        conn = get_db_connection() if
        not conn:
            return "Database unavailable", 500 cursor
        = conn.cursor(dictionary=True)
        reputation = get_reputation('producer', session['username']) cursor.execute('SELECT * FROM
        auctions WHERE
        initiator_type="producer" AND initiator_id=%s ORDER BY start_time DESC', (session['username'],))
        my_auctions = cursor.fetchall() cursor.execute('SELECT * FROM
        auctions WHERE
        initiator_type="procurer" AND status="active" ORDER BY start_time DESC') available_auctions = cursor.fetchall()
enhanced_available = []
        for auction in available_auctions:
            if auction['auction_type'] == 'open':
                cursor.execute('SELECT MAX(bid_amount) as highest_bid FROM bids WHERE
        auction_id=%s', (auction['auction_id'],))
                result = cursor.fetchone()
                auction['highest_bid'] = result['highest_bid'] if result['highest_bid'] else 'No bids yet'
            else:
                auction['highest_bid'] = 'Sealed'
            enhanced_available.append(auction)
        cursor.execute('''
            SELECT b.*, a.product_name, a.end_time, a.status as auction_status, a.initiator_type,
                   a.initiator_id, a.auction_type
            FROM bids b
            JOIN auctions a ON b.auction_id = a.auction_id
            WHERE b.bidder_id = %s AND b.bidder_type = 'producer'
            ORDER BY b.timestamp DESC ''',
        (session['username'],))
        my_bids = cursor.fetchall() cursor.execute('''
            SELECT * FROM auctions
            WHERE winner_id = %s AND status = 'closed'
            ORDER BY end_time DESC ''',
        (session['username'],)) won_auctions
        = cursor.fetchall()

        notifications = get_unread_notifications('producer', session['username']) cursor.close()
        conn.close()
        return render_template('web/producer_dashboard.html',
                               my_auctions=my_auctions,
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                                available_auctions=enhanced_available,
                                my_bids=my_bids, won_auctions=won_auctions,
                                notifications=notifications, reputation=reputation,
username=session['username']) @app.route('/procurer_dashboard')
            def procurer_dashboard():
                if 'username' not in session or session['role'] != 'procurer': return
                    redirect(url_for('login_prc'))
                conn = get_db_connection() if
                not conn:
                    return "Database unavailable", 500 cursor
                = conn.cursor(dictionary=True)

                reputation = get_reputation('procurer', session['username']) cursor.execute('SELECT * FROM
                auctions WHERE
            initiator_type="procurer" AND initiator_id=%s ORDER BY start_time DESC', (session['username'],))
                my_auctions = cursor.fetchall() cursor.execute('SELECT * FROM
                auctions WHERE
            initiator_type="producer" AND status="active" ORDER BY start_time DESC') available_auctions = cursor.fetchall()
                enhanced_available = []
                for auction in available_auctions:
                    if auction['auction_type'] == 'open':
                        cursor.execute('SELECT MAX(bid_amount) as highest_bid FROM bids WHERE
            auction_id=%s', (auction['auction_id'],))
                        result = cursor.fetchone()
                        auction['highest_bid'] = result['highest_bid'] if result['highest_bid'] else 'No bids yet'
                    else:
                        auction['highest_bid'] = 'Sealed'
                    enhanced_available.append(auction)
                cursor.execute('''
                    SELECT b.*, a.product_name, a.end_time, a.status as auction_status, a.initiator_type,
                           a.initiator_id, a.auction_type
                    FROM bids b
                    JOIN auctions a ON b.auction_id = a.auction_id
                    WHERE b.bidder_id = %s AND b.bidder_type = 'procurer'
                    ORDER BY b.timestamp DESC ''',
                (session['username'],))
my_bids = cursor.fetchall() cursor.execute('''
                    SELECT * FROM auctions
                    WHERE winner_id = %s AND status = 'closed'
                    ORDER BY end_time DESC ''',
                (session['username'],)) won_auctions
                = cursor.fetchall()
                notifications = get_unread_notifications('procurer', session['username']) cursor.close()
                conn.close()
                return render_template('web/procurer_dashboard.html',
                                my_auctions=my_auctions,
                                available_auctions=enhanced_available,
                                my_bids=my_bids, won_auctions=won_auctions,
                                notifications=notifications, reputation=reputation,
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                                    username=session['username'])
@app.route('/mark_notification_read/<int:notification_id>') def
mark_notification_read_route(notification_id):
    if 'username' not in session: return
        redirect(url_for('index'))
    mark_notification_read(notification_id)
    return redirect(request.referrer or url_for('index')) @app.route('/my_sealed_bids')
def my_sealed_bids():
    if 'username' not in session or session['role'] not in ['producer', 'procurer']: return
        redirect(url_for('index'))
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute('''
        SELECT b.*, a.product_name, a.end_time, a.status as auction_status FROM bids b
        JOIN auctions a ON b.auction_id = a.auction_id
        WHERE b.bidder_id = %s AND b.bidder_type = %s AND b.revealed = FALSE AND
a.auction_type = 'sealed'
        ORDER BY a.end_time DESC
''', (session['username'], session['role'])) bids = cursor.fetchall()
    cursor.close()
    conn.close()
    return render_template('web/my_sealed_bids.html', bids=bids)
@app.route('/reveal_my_bid/<int:bid_id>', methods=['POST']) def
reveal_my_bid(bid_id):
    if 'username' not in session: return
        redirect(url_for('index'))
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute('SELECT * FROM bids WHERE bid_id=%s', (bid_id,)) bid = cursor.fetchone()
    if not bid or bid['bidder_id'] != session['username'] or bid['bidder_type'] != session['role']:
        return "Not authorized", 403
    actual = request.form['actual_amount'] bid_data
    = json.loads(bid['bid_hash']) proof =
    bid_data['proof']  public_key_hex =
    bid_data['public_key']
    if verify_zk_proof(proof, public_key_hex):
        cursor.execute('UPDATE bids SET bid_amount=%s, revealed=TRUE WHERE bid_id=%s',
(actual, bid_id))
        conn.commit()
        add_notification(session['role'], session['username'], f"Your sealed bid for auction
{bid['auction_id']} has been revealed.", url_for('my_sealed_bids'))
        flash('Bid revealed successfully', 'success') else:
        flash('Invalid bid – zero-knowledge proof verification failed', 'danger') cursor.close()
    conn.close()
    return redirect(url_for('my_sealed_bids'))


@app.route('/admin_dashboard') def
admin_dashboard():
    if 'username' not in session or session['role'] != 'admin': return
        redirect(url_for('login_admin'))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
conn = get_db_connection() if not conn:
                    return "Database unavailable", 500 cursor
                = conn.cursor(dictionary=True)
                cursor.execute("SELECT producer_id, full_name, email FROM da_producer WHERE
            approved_status = 0")
                producers = cursor.fetchall()
                cursor.execute("SELECT procurer_id, full_name, email FROM da_procurer WHERE
            approved_status = 0")
                procurers = cursor.fetchall() cursor.execute('''
                    SELECT * FROM auctions
                    WHERE status = 'active' AND end_time < NOW() ORDER BY
                    end_time DESC
                ''')
                ended_auctions = cursor.fetchall()
                notifications = get_unread_notifications('admin', session['username']) cursor.close()
                conn.close()
                return render_template('web/admin_dashboard.html',
                                       producers=producers, procurers=procurers,
                                       ended_auctions=ended_auctions,
                                       notifications=notifications)
            @app.route('/approve/producer/<producer_id>') def
            approve_producer(producer_id):
                if 'username' not in session or session['role'] != 'admin': return
                    redirect(url_for('login_admin'))
                conn = get_db_connection() if
                not conn:
                    return "Database unavailable", 500 cursor
                = conn.cursor()
                cursor.execute('UPDATE da_producer SET approved_status=1 WHERE producer_id=%s', (producer_id,))
                conn.commit()
                cursor.close()
                conn.close()

add_block_to_chain(f"Producer approved: {producer_id}", "admin", session['username'])
                add_notification('producer', producer_id, 'Your account has been approved by admin. You can now
            log in.', url_for('login'))
                return redirect(url_for('admin_dashboard'))
            @app.route('/approve/procurer/<procurer_id>') def
            approve_procurer(procurer_id):
                if 'username' not in session or session['role'] != 'admin': return
                    redirect(url_for('login_admin'))
                conn = get_db_connection() if
                not conn:
                    return "Database unavailable", 500 cursor
                = conn.cursor()
                cursor.execute('UPDATE da_procurer SET approved_status=1 WHERE procurer_id=%s', (procurer_id,))
                conn.commit()
                cursor.close()
                conn.close()
                add_block_to_chain(f"Procurer approved: {procurer_id}", "admin", session['username'])
                add_notification('procurer', procurer_id, 'Your account has been approved by admin. You can now
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
log in.', url_for('login_prc'))
    return redirect(url_for('admin_dashboard'))
@app.route('/create_auction', methods=['POST']) def
create_auction():
    if 'username' not in session or session['role'] not in ['producer', 'procurer']: return
        redirect(url_for('index'))
    initiator_type = session['role'] initiator_id =
    session['username']
    product_name = request.form['product_name']
    description = request.form['description'] base_price =
    request.form['base_price'] auction_type =
    request.form['auction_type'] end_time =
    request.form['end_time']
   conn = get_db_connection() if
    not conn:
        return "Database unavailable", 500
cursor = conn.cursor() cursor.execute('''
        INSERT INTO auctions (initiator_type, initiator_id, product_name, description, base_price,
auction_type, start_time, end_time, status, payment_status, delivery_status)
        VALUES (%s, %s, %s, %s, %s, %s, NOW(), %s, 'active', 'pending', 'pending')
    ''', (initiator_type, initiator_id, product_name, description, base_price, auction_type, end_time))
    conn.commit()
    auction_id = cursor.lastrowid cursor.close()
    conn.close()
    add_block_to_chain(f"Auction created: {auction_id} by {initiator_id}", initiator_type, initiator_id)
    add_notification('admin', 'admin', f"New auction created by {initiator_type}
{initiator_id}: {product_name}", url_for('admin_dashboard')) if initiator_type
    == 'producer':
        return redirect(url_for('producer_dashboard')) else:
        return redirect(url_for('procurer_dashboard'))


@app.route('/place_bid/<int:auction_id>', methods=['POST']) def
place_bid(auction_id):
    if 'username' not in session or session['role'] not in ['producer', 'procurer']: return
        redirect(url_for('index'))
    bidder_type = session['role'] bidder_id
    = session['username'] conn =
    get_db_connection()
    if not conn:
        return "Database unavailable", 500 cursor
    = conn.cursor(dictionary=True)
    cursor.execute('SELECT * FROM auctions WHERE auction_id=%s', (auction_id,))
    auction = cursor.fetchone()
    if not auction or auction['status'] != 'active': flash('Auction not
        active', 'danger')
return redirect(url_for('producer_dashboard' if bidder_type=='producer' else 'procurer_dashboard'))
    if auction['initiator_type'] == bidder_type:
        flash('You cannot bid on your own auction', 'danger')
        return redirect(url_for('producer_dashboard' if bidder_type=='producer' else
'procurer_dashboard'))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    bid_amount = request.form['bid_amount'] if
    auction['auction_type'] == 'sealed':
        nonce = os.urandom(16).hex()
        proof, public_key = generate_zk_proof(bid_amount, nonce) combined =
        json.dumps({'proof': proof, 'public_key': public_key}) cursor.execute('''
            INSERT INTO bids (auction_id, bidder_type, bidder_id, bid_amount, bid_hash, revealed,
timestamp)
            VALUES (%s, %s, %s, %s, %s, %s, NOW())
        ''', (auction_id, bidder_type, bidder_id, 0, combined, False)) else:
         ring_keys = []
        if bidder_type == 'producer':
            cursor.execute('SELECT public_key FROM da_procurer WHERE public_key IS NOT
NULL')
        else:
            cursor.execute('SELECT public_key FROM da_producer WHERE public_key IS NOT
NULL')
        rows = cursor.fetchall() if
        rows:
            ring_keys = [row['public_key'] for row in rows] else:
            key = generate_rsa_keypair()
            ring_keys = [key.publickey().export_key().decode('utf-8')] signer_key =
        generate_rsa_keypair()
        message = f"{auction_id}:{bidder_id}:{bid_amount}".encode() ring_sig =
        ring_sign(ring_keys, signer_key, message)  combined = json.dumps(ring_sig)
        cursor.execute('''
            INSERT INTO bids (auction_id, bidder_type, bidder_id, bid_amount, bid_hash, timestamp)

VALUES (%s, %s, %s, %s, %s, NOW())
        ''', (auction_id, bidder_type, bidder_id, bid_amount, combined)) conn.commit()
    cursor.close()
    conn.close()
    add_block_to_chain(f"{auction['auction_type'].capitalize()} bid placed in auction {auction_id} by
{bidder_id}", bidder_type, bidder_id)
    add_notification(auction['initiator_type'], auction['initiator_id'], f"A new bid was placed on your
auction {auction_id} ({auction['product_name']}).", url_for('producer_dashboard' if
auction['initiator_type']=='producer' else 'procurer_dashboard'))
    flash('Bid placed successfully', 'success')
    return redirect(url_for('producer_dashboard' if bidder_type=='producer' else 'procurer_dashboard'))
@app.route('/close_auction/<int:auction_id>', methods=['POST']) def
close_auction(auction_id):
    if 'username' not in session or session['role'] != 'admin': flash('You need to
        be logged in as admin.', 'danger') return
        redirect(url_for('login_admin'))
    conn = get_db_connection() if
    not conn:
        flash('Database unavailable.', 'danger') return
        redirect(url_for('admin_dashboard'))
    cursor = conn.cursor(dictionary=True)
    cursor.execute('SELECT * FROM auctions WHERE auction_id=%s', (auction_id,))
    auction = cursor.fetchone()
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    if not auction or auction['status'] != 'active': flash('Auction not active
        or does not exist.', 'warning') return
        redirect(url_for('admin_dashboard'))
    if auction['auction_type'] == 'sealed':
        cursor.execute('SELECT * FROM bids WHERE auction_id=%s AND revealed=TRUE', (auction_id,))
    else:
        cursor.execute('SELECT * FROM bids WHERE auction_id=%s', (auction_id,))
    bids = cursor.fetchall()
bids = [b for b in bids if b['bidder_id'] != auction['initiator_id']] if not bids:
        cursor.execute('UPDATE auctions SET status="closed", winner_id=NULL, final_price=NULL
WHERE auction_id=%s', (auction_id,))
        conn.commit
        ()
        cursor.close(
        )
        conn.close()
        add_block_to_chain(f"Auction {auction_id} closed with no bids", "admin",
session['username'])
        add_notification(auction['initiator_type'], auction['initiator_id'], f"Your auction {auction_id}
closed with no bids.", url_for('producer_dashboard' if auction['initiator_type']=='producer' else
'procurer_dashboard'))
        flash('Auction closed with no bids.', 'info') return
        redirect(url_for('admin_dashboard'))
    if auction['initiator_type'] == 'producer':
        winner = max(bids, key=lambda b: b['bid_amount']) else:
        winner = min(bids, key=lambda b: b['bid_amount']) cursor.execute('UPDATE auctions SET
    status="closed", winner_id=%s,
final_price=%s WHERE auction_id=%s',
                (winner['bidder_id'], winner['bid_amount'], auction_id)) conn.commit()
    add_reputation(winner['bidder_type'], winner['bidder_id'], 10) cursor.close()
    conn.close()
    add_block_to_chain(f"Auction {auction_id} closed, winner:
    {winner['bidder_id']}", "admin", session['username']) add_notification(winner['bidder_type'], winner['bidder_id'],
    f"Congratulations! You won auction {auction_id} ({auction['product_name']}).",
    url_for('producer_dashboard' if winner['bidder_type']=='producer' else
    'procurer_dashboard'))
    add_notification(auction['initiator_type'], auction['initiator_id'], f"Your auction {auction_id}
    closed. Winner: {winner['bidder_id']} with bid
    {winner['bid_amount']}.", url_for('producer_dashboard' if auction['initiator_type']=='producer' else
    'procurer_dashboard'))
    flash('Auction closed successfully.', 'success') return
    redirect(url_for('admin_dashboard'))
@app.route('/make_payment/<int:auction_id>') def
make_payment(auction_id):
    if 'username' not in session: return
        redirect(url_for('index'))
    conn = get_db_connection() if
    not conn:
        flash('Database error', 'danger')
        return redirect(url_for('index'))
    cursor = conn.cursor(dictionary=True)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    cursor.execute('SELECT * FROM auctions WHERE auction_id=%s', (auction_id,))
    auction = cursor.fetchone()
    if not auction or auction['winner_id'] != session['username']: flash('Not authorized',
        'danger')
        return redirect(url_for('index'))
    if auction['payment_status'] == 'paid': flash('Payment
        already completed', 'info') return
        redirect(request.referrer)
    cursor.execute('UPDATE auctions SET payment_status="paid", delivery_status="delivered" WHERE
auction_id=%s', (auction_id,))
    conn.commit()
    add_reputation(session['role'], session['username'], 10) add_block_to_chain(f"Payment and delivery
    completed for auction
{auction_id} by {session['username']}", session['role'], session['username']) add_notification(session['role'],
    session['username'], f"Payment of
{auction['final_price']} completed for auction {auction_id}. +10 reputation.",
url_for('producer_dashboard' if session['role']=='producer' else 'procurer_dashboard'))
    cursor.close()
    conn.close()
    flash(f'Payment of {auction["final_price"]} processed successfully! +10 reputation.', 'success')
    return redirect(request.referrer) @app.route('/view_block')
def view_block():
    if 'username' not in session or session['role'] != 'admin': return
        redirect(url_for('login_admin'))
try:
        with open('dualchain.json', 'r') as f: chain =
            json.load(f)
        for block in chain:
            if 'hash' not in block: block_copy
                = block.copy() if 'hash' in
                block_copy:
                    del block_copy['hash']
                block_string = json.dumps(block_copy, sort_keys=True).encode() block['hash'] =
                hashlib.sha256(block_string).hexdigest()
        with open('dualchain.json', 'w') as f: json.dump(chain, f,
            indent=4)
    except FileNotFoundError:
        chain = []
    return render_template('web/view_block.html', chain=chain) @app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index')) if _name_____== '____main___':
    print(" Server starting on http://localhost:8080") app.run(debug=True, host='0.0.0.0',
    port=8080, use_reloader=False)
```

HTML

base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dual-BlockBid - {% block title %}{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="d-flex flex-column min-vh-100">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">Dual-BlockBid</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav">
                <span  class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    {% if session.username %}
                        {% if session.role == 'producer' %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('producer_dashboard')
}}">Dashboard</a>
                </li>
                        {% elif session.role == 'procurer' %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('procurer_dashboard')
}}">Dashboard</a>
                </li>

                        {% elif session.role == 'admin' %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('admin_dashboard')

}}">Admin Panel</a>
                        </li>
                        {% endif %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('logout') }}"> Logout ({{
                                session.username }})
                            </a>
                        </li>
                    {% else %}
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" data-bs- toggle="dropdown">
                                Login
                            </a>
                            <ul class="dropdown-menu">
                                <li><a class="dropdown-item" href="{{ url_for('login')
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
}}">Producer</a></li>
                                <li><a class="dropdown-item" href="{{ url_for('login_prc')
}}">Procurer</a></li>
                                <li><a class="dropdown-item" href="{{ url_for('login_admin')
}}">Admin</a></li>
                            </ul>
                        </li>
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle" href="#" data-bs- toggle="dropdown">
                                Register
                            </a>
                            <ul class="dropdown-menu">
                                <li><a class="dropdown-item" href="{{ url_for('register_producer')
}}">Producer</a></li>
                                <li><a class="dropdown-item" href="{{ url_for('register_procurer')
}}">Procurer</a></li>
                            </ul>
                        </li>
                    {% endif %}
                </ul>
</div>
            </div>
        </nav>
        <!-- Main Content -->
        <main class="container my-4 flex-grow-1">
            <!-- Flash messages -->
            {% with messages = get_flashed_messages(with_categories=true) %}
              {% if messages %}
                {% for category, message in messages %}
                  <div class="alert alert-{{ category }} alert-dismissible fade show">
                    {{ message }}
                    <button type="button" class="btn-close" data-bs- dismiss="alert"></button>
                  </div>
                {% endfor %}
              {% endif %}
            {% endwith %}
            {% block content %}{% endblock %}
        </main>
        <!-- Footer -->
        <footer class="bg-dark text-white text-center py-3 mt-auto">
            <p class="mb-0">&copy; 2026 Dual-BlockBid – Blockchain Auction System</p>
        </footer>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j s"></script>
    </body>
    </html>
```

index.html

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
{% extends "web/base.html" %}
{% block title %}Home{% endblock %}
{% block content %}
<div class="text-center">
```
```
<img src="{{ url_for('static', filename='images/auction_banner.png') }}" class="img-fluid
     rounded shadow mb-4"
     alt="Auction Banner">
<h1 class="display-4">Welcome to Dual-BlockBid</h1>
<p class="lead">A secure, dual-mode blockchain auction system for multi- category products.</p>
<hr class="my-4">
<p>
    Producers and procurers can trade agricultural goods, artworks, home appliances,
    and more with full privacy and transparency.
</p>
```
```
</div>
{% endblock %}
```

Register_Producer.html

```
{% extends "web/base.html" %}
{% block title %}Producer Registration{% endblock %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <h2>Producer Registration</h2>
        <form method="post">
            <div class="mb-3">
                <label>Producer ID</label>
                <input type="text" name="producer_id" class="form-control"
required>
            </div>
            <div class="mb-3">
                <label>Password</label>
                <input type="password" name="password" class="form-control"
required>
            </div>
            <div class="mb-3">
                <label>Company Name</label>
                <input type="text" name="company" class="form-control" required>
            </div>
            <div class="mb-3">
                <label>Full Name</label>
                <input type="text" name="fullname" class="form-control" required>
            </div>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                    <div class="mb-3">
                        <label>Mobile</label>
                        <input type="text" name="mobile" class="form-control" required>
                    </div>
                    <div class="mb-3">
                        <label>Identity Proof</label>
                        <input type="text" name="proof" class="form-control" placeholder="e.g., Aadhar,
            Passport" required>
                    </div>
                    <div class="mb-3">
                        <label>Email</label>
<input type="email" name="email" class="form-control" required>
                    </div>
                    <button type="submit" class="btn btn-primary">Register</button>
                </form>
            </div>
        </div>
        {% endblock %}
```

login_prc.html

```
        {% extends "web/base.html" %}
        {% block title %}Procurer Login{% endblock %}
        {% block content %}
        <div class="row justify-content-center">
            <div class="col-md-6">
                <h2>Procurer Login</h2>
                {% if msg %}<div class="alert alert-danger">{{ msg }}</div>{% endif
        %}
                 <form method="post">
                    <div class="mb-3">
                        <label>Username</label>
                        <input type="text" name="uname" class="form-control" required>
                    </div>
                    <div class="mb-3">
                        <label>Password</label>
                        <input type="password" name="pass" class="form-control" required>
                    </div>
                    <button type="submit" class="btn btn-primary">Login</button>
                </form>
            </div>
        </div>
        {% endblock %}
```

procurer_dashboard.html

```
        {% extends "web/base.html" %}
        {% block title %}Procurer Dashboard{% endblock %}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
{% block content %}
<h2>Procurer Dashboard</h2>
<p><strong>Your Reputation Score:</strong> {{ reputation }}</p>
<!-- Notifications Panel -->
<div class="card mb-4">
    <div class="card-header bg-info text-white">
        Notifications {% if notifications %}<span class="badge bg-danger">{{ notifications|length }}</span>{%
endif %}
    </div>
     <div class="card-body">
        {% if notifications %}
            <ul class="list-group">
                {% for notif in notifications %}
                <li class="list-group-item d-flex justify-content-between align-items-

center">

{{ notif.message }}
<small>{{ notif.created_at }}</small>
<a href="{{ notif.link }}" class="btn btn-sm btn-


primary">View</a>
                        <a href="{{ url_for('mark_notification_read_route', notification_id=notif.id) }}"
class="btn btn-sm btn-secondary">Mark Read</a>
                </li>
                {% endfor %}
            </ul>
        {% else %}
            <p class="text-muted">No new notifications.</p>
        {% endif %}
    </div>
</div>
<div class="row">
    <div class="col-md-6">
        <h3>My Auctions (Initiated by me)</h3>
        {% if my_auctions %}
            <table class="table table-striped">


                <thead>
                    <tr>
                        <th>Product</th>
                        <th>Type</th>
                        <th>End Time</th>
                        <th>Status</th>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                        <th>Payment</th>
                        <th>Delivery</th>
                    </tr>
                </thead>
                <tbody>
                {% for auc in my_auctions %}
                    <tr>
                        <td>{{ auc.product_name }}</td>
                        <td>{{ auc.auction_type }}</td>
                        <td>{{ auc.end_time }}</td>
                        <td>{{ auc.status }}</td>
                        <td>{{ auc.payment_status }}</td>
                        <td>{{ auc.delivery_status }}</td>
                    </tr>
                {% endfor %}
                </tbody>
            </table>
        {% else %}
            <p>No auctions created yet.</p>
        {% endif %}
        <h4>Request New Auction</h4>
        <form method="POST" action="{{ url_for('create_auction') }}">
            <div class="mb-2">
                <input type="text" name="product_name" placeholder="Product Name" class="form-
control" required>
            </div>
            <div class="mb-2">
                <textarea name="description" placeholder="Description" class="form-
control"></textarea>
            </div>
            <div class="mb-2">
<input type="number" step="0.01" name="base_price" placeholder="Base Price" class="form-control" required>
            </div>
            <div class="mb-2">
                <select name="auction_type" class="form-control">
                    <option value="open">Open</option>
                    <option value="sealed">Sealed</option>
                </select>
            </div>
            <div class="mb-2">
                <input type="datetime-local" name="end_time" class="form-control"
required>
            </div>
            <button type="submit" class="btn btn-primary">Request Auction</button>
        </form>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    </div>
    <div class="col-md-6">
        <h3>Available Auctions (to bid on)</h3>
        {% if available_auctions %}
            <table class="table table-striped">
                <thead><tr><th>Product</th><th>Type</th><th>Base
Price</th><th>Best Bid</th><th>End Time</th><th>Action</th></tr></thead>
                <tbody>
                {% for auc in available_auctions %}
                    <tr>
                        <td>{{ auc.product_name }}</td>
                        <td>{{ auc.auction_type }}</td>
                        <td>{{ auc.base_price }}</td>
                        <td>{{ auc.highest_bid }}</td>
                        <td>{{ auc.end_time }}</td>
                        <td>
                            <form method="POST" action="{{ url_for('place_bid', auction_id=auc.auction_id) }}"
style="display:inline;">
                                <input type="number" step="0.01" name="bid_amount" placeholder="Bid" required>

                                <button type="submit" class="btn btn-sm btn- success">Bid</button>
                            </form>
                        </td>
                    </tr>
                {% endfor %}
                </tbody>
        </table>
    {% else %}
            <p>No auctions available.</p>
        {% endif %}
    </div>
</div>
<div class="row mt-4">
    <div class="col-md-6">
        <h3>My Bids</h3>
        {% if my_bids %}
            <table class="table table-sm">
            <thead>
                <tr>
                    <th>Auction</th>
                    <th>Bid Amount</th>
                    <th>Revealed</th>
                    <th>Status</th>
                </tr>
            </thead>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                <tbody>
                {% for bid in my_bids %}
                    <tr>
                        <td>{{ bid.product_name }}</td>
                        <td>
                            {% if bid.auction_type == 'sealed' and not bid.revealed %} Hidden
                            {% else %}
                                {{ bid.bid_amount }}
                            {% endif %}
                        </td>

                        <td>
                                    {% if bid.auction_type == 'sealed' %}
                                      {{ 'Yes' if bid.revealed else 'No' }}
                            {% else
                                %}
                                N/A
                            {% endif %}
                        </td>
                        <td>{{ bid.auction_status }}</td>
                    </tr>
                {% endfor %}
                </tbody>
            </table>
            <a href="{{ url_for('my_sealed_bids') }}" class="btn btn-warning btn- sm">Reveal Sealed
Bids</a>
        {% else %}
            <p>No bids placed.</p>
        {% endif %}
    </div>

    <div class="col-md-6">
        <h3>Auctions I Won</h3>
        {% if won_auctions %}
            <table class="table table-sm">
                <thead>
                    <tr>
                        <th>Product</th>
                        <th>Final Price</th>
                        <th>Payment</th>
                        <th>Delivery</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
                    {% for auc in won_auctions %}
                        <tr>
                            <td>{{ auc.product_name }}</td>
                            <td>{{ auc.final_price }}</td>

                            <td>{{ auc.payment_status }}</td>
                            <td>{{ auc.delivery_status }}</td>
                            <td>
                                {% if auc.payment_status == 'pending' %}
                                    <a href="{{ url_for('make_payment', auction_id=auc.auction_id) }}"
class="btn btn-sm btn-primary">Pay Now</a>
                                {% else %}
                                    <span class="text-success">Payment Success &
Delivered</span>
                                {% endif %}
                            </td>
                        </tr>
                    {% endfor %}
                    </tbody>
                </table>
            {% else %}
                    <p>You haven't won any auctions yet.</p>
                {% endif %}
            </div>
        </div>
        {% endblock %}
```

admin_dashboard.html

```
        {% extends "web/base.html" %}
        {% block title %}Admin Dashboard{% endblock %}
        {% block content %}
        <h2>Admin Dashboard</h2>
        <!-- Notifications Panel -->
        <div class="card mb-4">
            <div class="card-header bg-info text-white">
                Notifications {% if notifications %}<span class="badge bg-danger">{{ notifications|length }}</span>{%
endif %}
            </div>
              <div class="card-body">
                  {% if notifications %}
                      <ul class="list-group">
                          {% for notif in notifications %}
                          <li class="list-group-item d-flex justify-content-between align-items-

center">
{{ notif.message }}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
<small>{{ notif.created_at }}</small>
<a href="{{ notif.link }}" class="btn btn-sm btn-

primary">View</a>
                    <a href="{{ url_for('mark_notification_read_route', notification_id=notif.id) }}"
class="btn btn-sm btn-secondary">Mark Read</a>
                </li>
                {% endfor %}
            </ul>
        {% else %}
            <p class="text-muted">No new notifications.</p>
        {% endif %}
    </div>
</div>
<!-- Pending Approvals -->
<div class="row">
    <div class="col-md-6">
        <h3>Pending Producers</h3>
        <ul class="list-group">
            {% for p in producers %}

            <li class="list-group-item d-flex justify-content-between">
                {{ p.full_name }} ({{ p.email }})
                <a href="{{ url_for('approve_producer', producer_id=p.producer_id)
}}" class="btn btn-sm btn-success">Approve</a>
            </li>
            {% else %}
            <li class="list-group-item">No pending producers.</li>
            {% endfor %}
        </ul>
    </div>
    <div class="col-md-6">
        <h3>Pending Procurers</h3>
        <ul class="list-group">
            {% for p in procurers %}
            <li class="list-group-item d-flex justify-content-between">
                {{ p.full_name }} ({{ p.email }})
                <a href="{{ url_for('approve_procurer', procurer_id=p.procurer_id)
}}" class="btn btn-sm btn-success">Approve</a>
            </li>
            {% else %}
            <li class="list-group-item">No pending procurers.</li>
            {% endfor %}
        </ul>
    </div>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
</div>
<!-- Ended Auctions (Ready to Close) -->
<h3 class="mt-4">Ended Auctions (Ready to Close)</h3>
<table class="table table-striped">
    <thead>
        <tr>
            <th>Auction ID</th>
            <th>Product</th>
            <th>Initiator</th>
            <th>Type</th>
            <th>End Time</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        {% for a in ended_auctions %}
        <tr>
            <td>{{ a.auction_id }}</td>
            <td>{{ a.product_name }}</td>
            <td>{{ a.initiator_id }} ({{ a.initiator_type }})</td>
            <td>{{ a.auction_type }}</td>
            <td>{{ a.end_time }}</td>
            <td>
                <form action="{{ url_for('close_auction', auction_id=a.auction_id)
}}" method="post" style="display:inline;">
                    <button type="submit" class="btn btn-sm btn-danger">Close Auction</button>
                </form>
                {% if a.auction_type == 'sealed' %}
                    <small class="text-muted">(Ensure all bids are revealed first)</small>
                {% endif %}
            </td>
        </tr>
        {% else %}
        <tr><td colspan="6">No ended auctions.</td></tr>
        {% endfor %}
    </tbody>
</table>
<div class="mt-3">
    <a href="{{ url_for('view_block') }}" class="btn btn-primary">View Blockchain Integrity</a>
</div>
{% endblock %}
```

reveal_bid.html

```
{% extends "web/base.html" %}
{% block title %}Reveal Bids{% endblock %}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
{% block content %}
<h2>Reveal Bids for Auction #{{ auction.auction_id }} – {{ auction.product_name }}</h2>
<form action="{{ url_for('reveal_bids_submit', auction_id=auction.auction_id)
}}" method="post">
    <table class="table table-bordered">
        <thead>
            <tr><th>Bidder ID</th><th>Commitment</th><th>Actual Bid Amount</th></tr>
        </thead>
        <tbody>
            {% for bid in bids %}
            <tr>                    <td>{{ bid.bidder_id }}</td>
                <td><code>{{ bid.bid_hash }}</code></td>
                <td><input type="number" step="0.01" name="actual_{{ bid.bid_id
}}" class="form-control" required></td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    <button type="submit" class="btn btn-primary">Submit Revealed Bids</button>
    <a href="{{ url_for('admin_dashboard') }}" class="btn btn- secondary">Cancel</a>
</form>
{% endblock %}
```

view_block.html

```
{% extends "web/base.html" %}
{% block title %}Blockchain Viewer{% endblock %}
{% block content %}
<h2>Blockchain Integrity</h2>
<div class="mb-3">
    <a href="{{ url_for('admin_dashboard') }}" class="btn btn-secondary">Back to Dashboard</a>
</div>
<div class="accordion" id="blockchainAccordion">
    {% for block in chain %}
    <div class="accordion-item">
        <h2 class="accordion-header" id="heading{{ block.index }}">
            <button class="accordion-button collapsed" type="button" data-bs- toggle="collapse" data-bs-
target="#collapse{{ block.index }}">
                Block #{{ block.index }} – {{ block.timestamp }}
            </button>
        </h2>
        <div id="collapse{{ block.index }}" class="accordion-collapse collapse" data-bs-
parent="#blockchainAccordion">
            <div class="accordion-body">
                <pre>{{ block | tojson(indent=2) }}</pre>
            </div>
        </div>
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
    </div>
    {% endfor %}
</div>
{% endblock %}
```

DATABASE SCHEMA (MySQL)

```
CREATE DATABASE IF NOT EXISTS double_auction_dual; USE
double_auction_dual;
CREATE TABLE da_admin (
    username VARCHAR(50) PRIMARY KEY,
    password VARCHAR(255) NOT NULL                    );
CREATE TABLE da_producer (
    producer_id VARCHAR(50) PRIMARY KEY, password
    VARCHAR(255) NOT NULL,
    company_name VARCHAR(100), full_name
    VARCHAR(100) NOT NULL, mobile
    VARCHAR(20),
    proof VARCHAR(255), email
    VARCHAR(100),
    reputation_score INT DEFAULT 0,
    public_key TEXT,
    approved_status TINYINT(1) DEFAULT 0
);
CREATE TABLE da_procurer (
    procurer_id VARCHAR(50) PRIMARY KEY, password
    VARCHAR(255) NOT NULL,
    full_name VARCHAR(100) NOT NULL, mobile
    VARCHAR(20),
    proof VARCHAR(255), email
    VARCHAR(100),
    reputation_score INT DEFAULT 0,
    public_key TEXT,
    approved_status TINYINT(1) DEFAULT 0
);
CREATE TABLE auctions (
    auction_id INT AUTO_INCREMENT PRIMARY KEY,
    initiator_type ENUM('producer','procurer') NOT NULL, initiator_id
    VARCHAR(50) NOT NULL,  product_name VARCHAR(255) NOT
    NULL, description TEXT,
    base_price DECIMAL(10,2) NOT NULL, auction_type
    ENUM('open','sealed') NOT NULL,
    start_time DATETIME NOT NULL,
    end_time DATETIME NOT NULL,
    status ENUM('active','closed') DEFAULT 'active', winner_id
    VARCHAR(50),
    final_price DECIMAL(10,2),
    payment_status ENUM('pending','paid') DEFAULT 'pending', delivery_status
    ENUM('pending','delivered') DEFAULT 'pending', INDEX idx_initiator
    (initiator_id),
    INDEX idx_status (status)
```

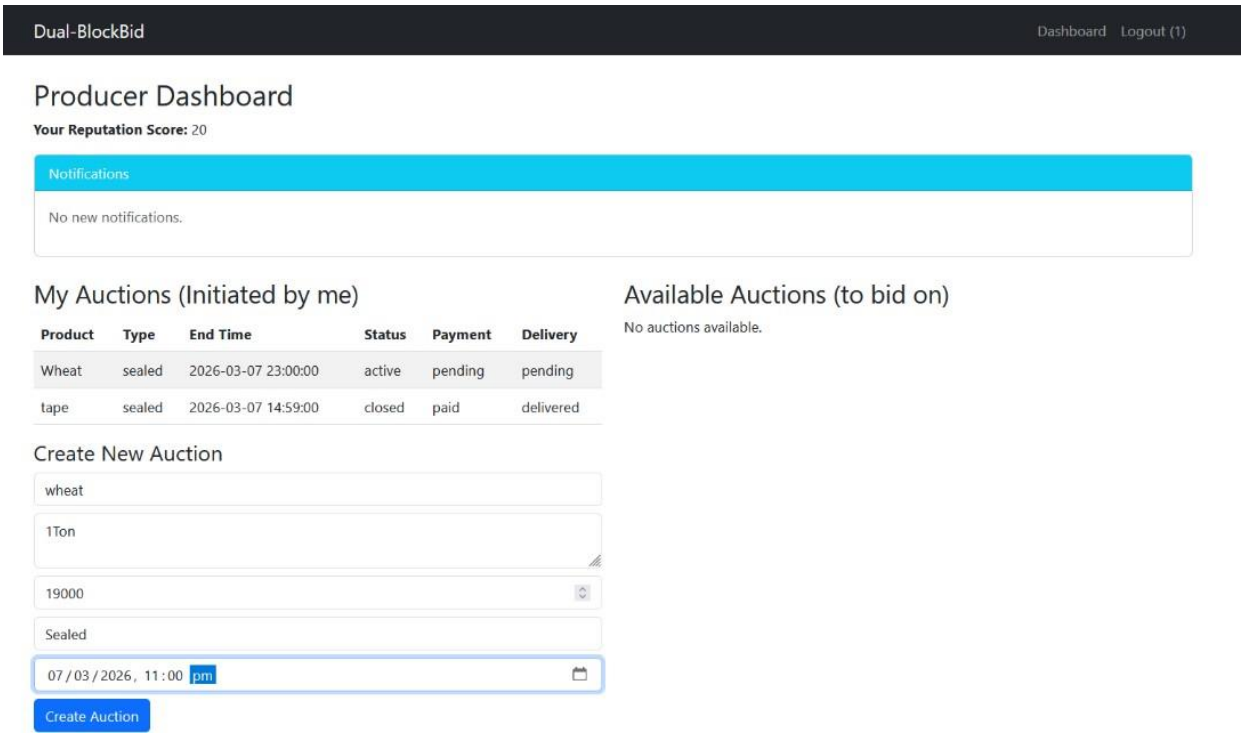
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

```
);
CREATE TABLE bids (
    bid_id INT AUTO_INCREMENT PRIMARY KEY,
    auction_id INT NOT NULL,
    bidder_type ENUM('producer','procurer') NOT NULL, bidder_id
    VARCHAR(50) NOT NULL,
    bid_amount DECIMAL(10,2) NOT NULL, bid_hash
    TEXT,
    revealed TINYINT(1) DEFAULT FALSE,
    timestamp DATETIME NOT NULL,
    FOREIGN KEY (auction_id) REFERENCES auctions(auction_id) ON DELETE CASCADE,
    INDEX idx_auction (auction_id), INDEX
    idx_bidder (bidder_id)
);
CREATE TABLE notifications (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_type ENUM('producer','procurer','admin') NOT NULL, user_id
    VARCHAR(50) NOT NULL,
    message TEXT NOT NULL, link
    VARCHAR(255),
    created_at DATETIME NOT NULL, is_read
    TINYINT(1) DEFAULT FALSE,
    INDEX idx_user (user_type, user_id)
);
```

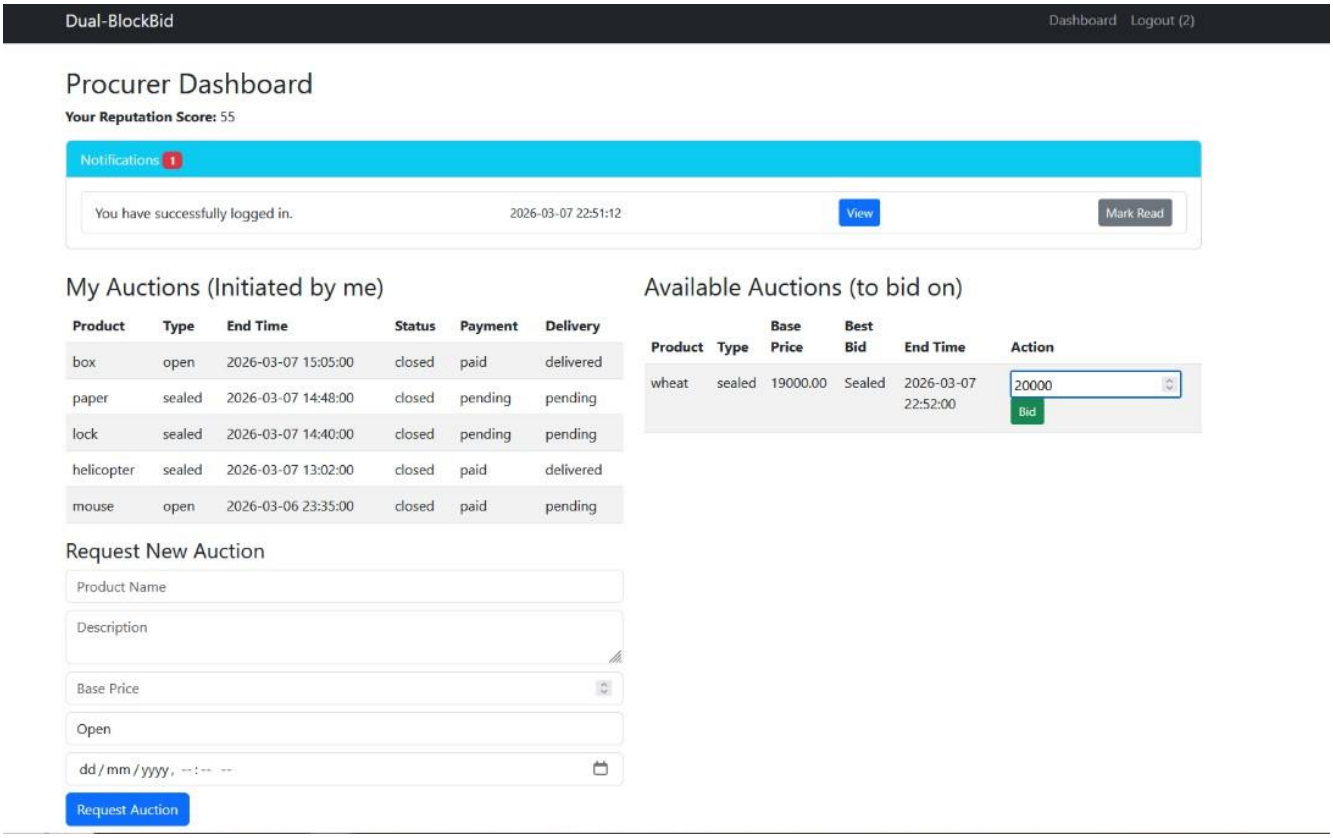
APPENDIX – II SCREENSHOTS

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Homepage

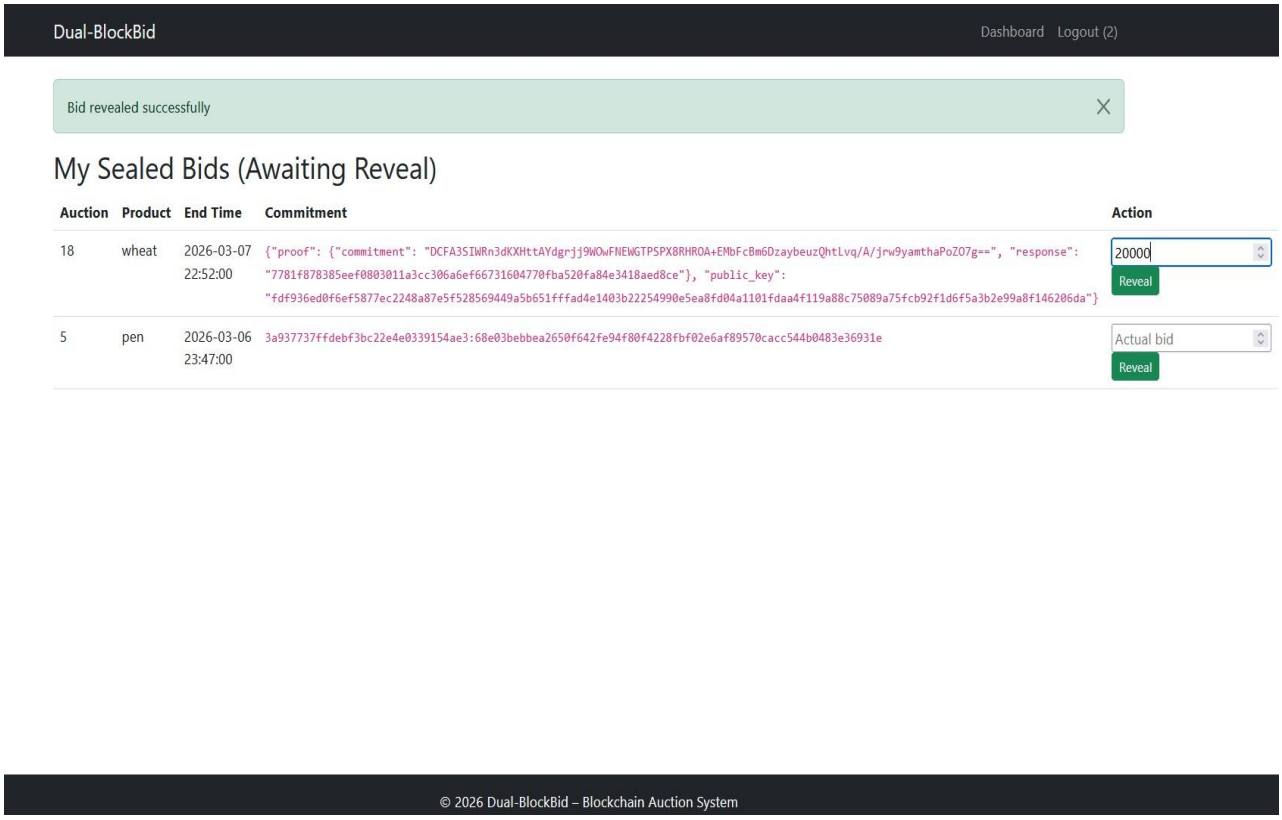


Producer Dashboard – Create Aution

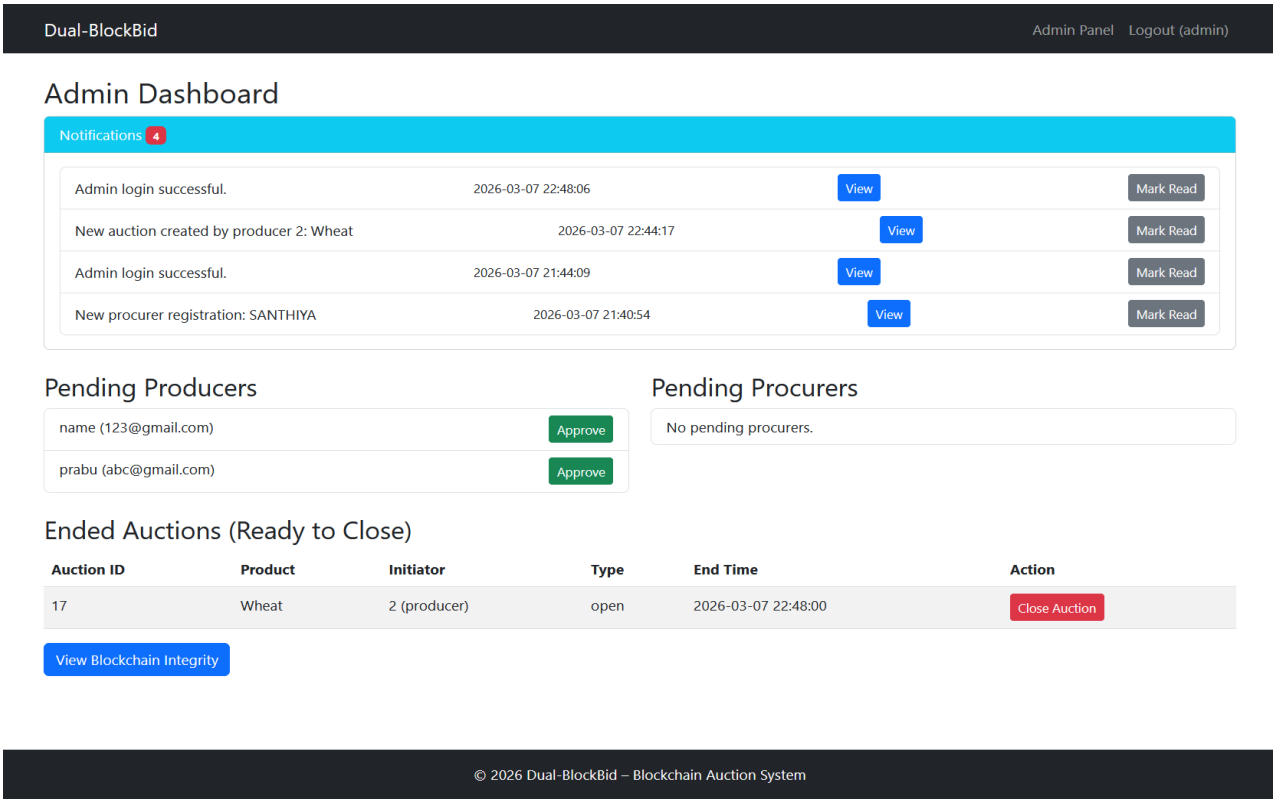


International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Procurer Dashboard – Auction to bid on

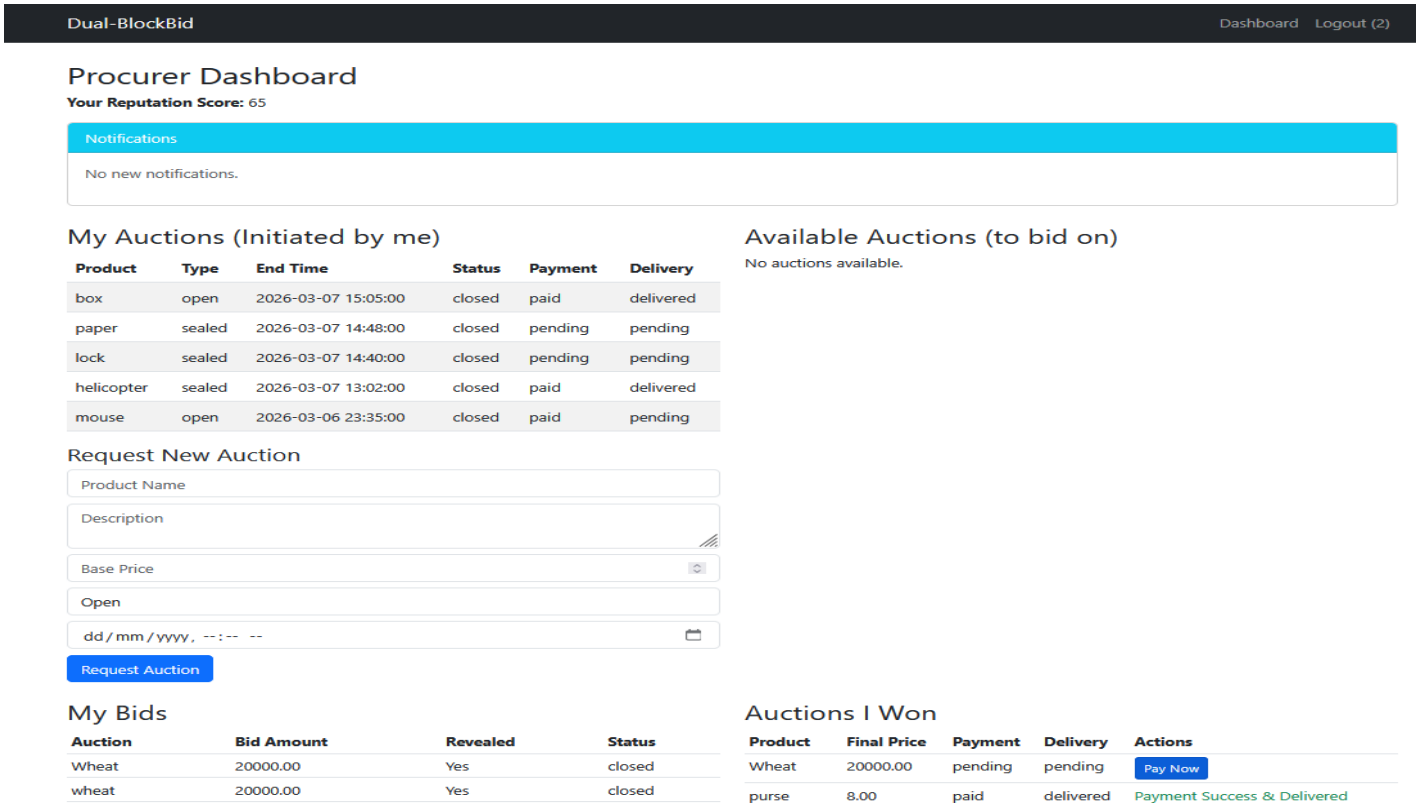


Sealed Bid Reveal

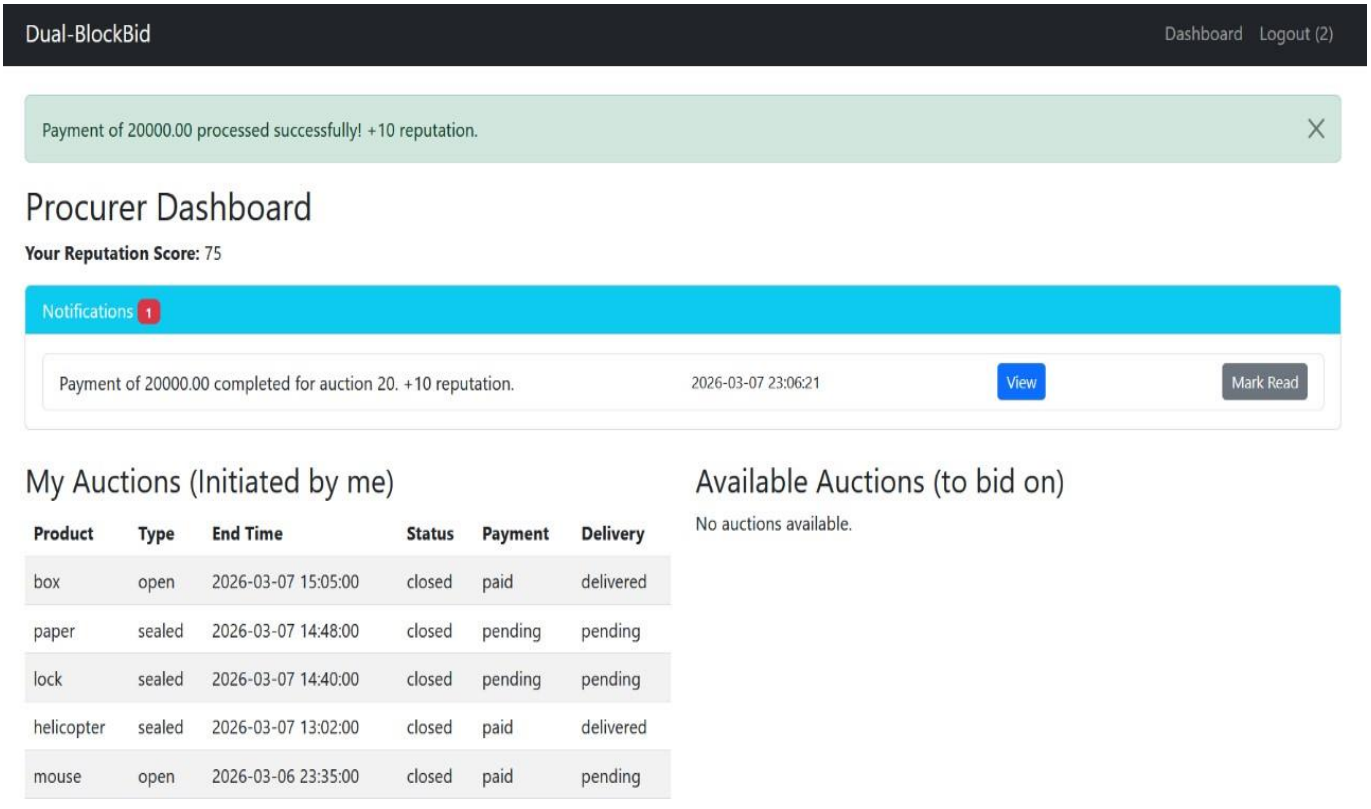


International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Admin Dashboard- Close Auction

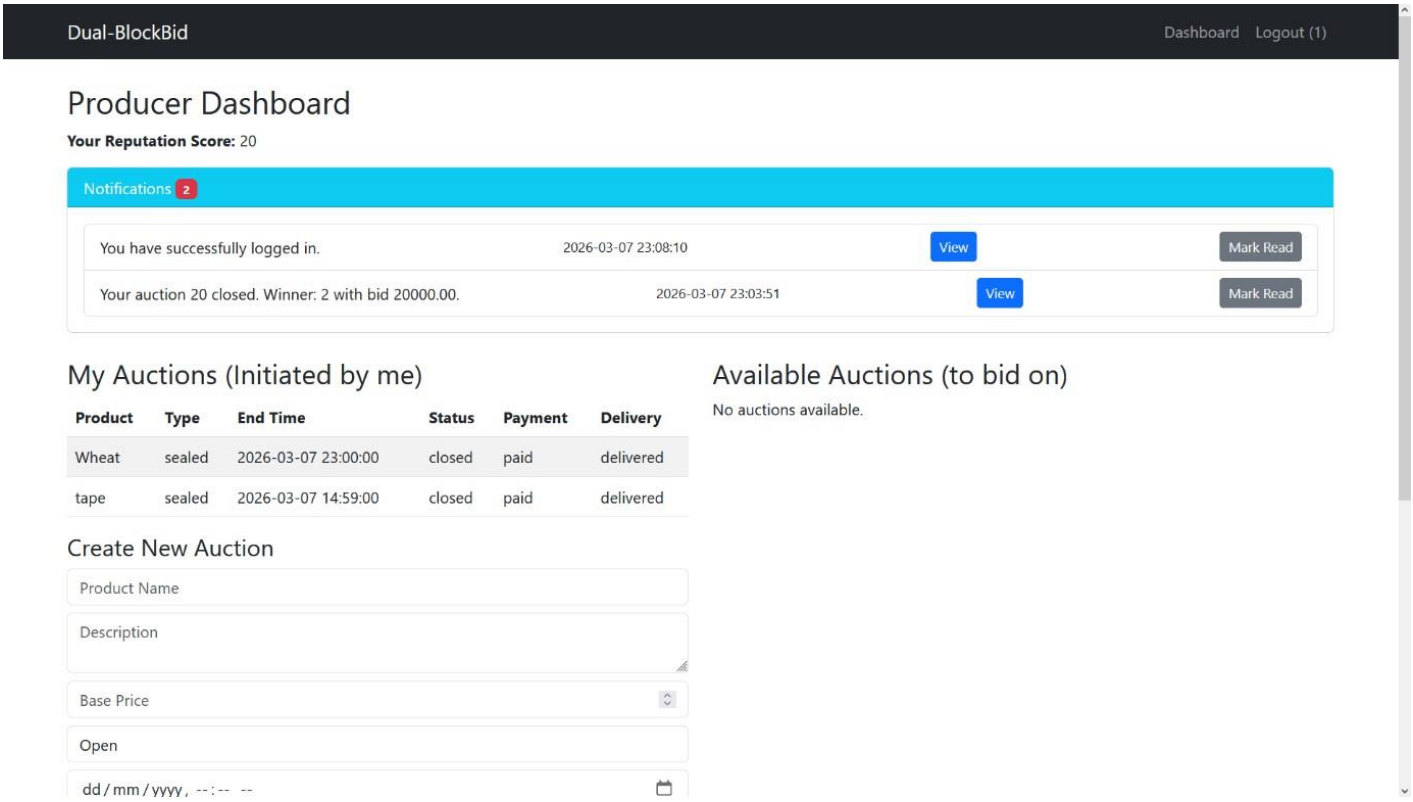


Procurer Dashboard – Procurer Win the Auction



Procurer Dashboard – Procurer Payment and Reputation Score

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026



Producer Dashboard - Producer Received the payment and Delivered

REFERENCES

[1] Arunkumar Chincheti, Rashmi Kamlakar Dixit, Vijay Anant Athavale1, L.
M. R. J. Lobo, Maria Lapina, “A Safe and Secure Online System for Bidding Using Blockchain Technology”, Walchand Institute of Technology, North-Caucasus University, Oct 2024.

[2] Burhan. M, Mustafa. G, Rana. M. R. R, Shaukat. R. S, Abbas. G, “A Blockchain Based Framework for Secure Public and Sealed-Bid Auctions with AES Encryption”, IJIST, Vol. 06 Issue. 04 pp 2140-2158, Dec 2024.

[3] Chetana Nemade, Saurav Borane, Sandeep Sharma, Akash Borase,” Online Auction System Based on Distributed Technology”, International Journal Of Science, Spirituality, Business And Technology (IJSSBT), Vol. 2, No.1, November 2023.

[4] Nafiseh Sharghivand, Farnaz Derakhshan, “Comprehensive Survey on Auction Mechanism Design for Cloud/Edge Resource Management and Pricing”, IEEE Access paper, doi. 10.1109/ACCESS.2021.3110914, Sep 2021.

[5] Sativa Tripathi, Rashik Walia, Dr. Kapil Rana, “Secure Online Auction System”, Jaypee University of Information Technology, 2023-2024.

[6] Yang Zhang, Huibing Cheng, “Novel Double Auction Mechanisms for. Agricultural Supply Chain Trading”, IEEE Access, Oct 2023.

[7] Zhang. Q, Yu. Y, Li. H, Yu. J and Wang. L, "Trustworthy sealed-bid auction with low communication cost atop blockchain", Inf. Sci., vol.631, pp. 202-217, Jun. 2023.

[8] “Privacy-Preserving Tools and Technologies”: Government Adoption and Challenges SIDIK PRABOWO 1, (Student Member, IEEE), AJI

[9] GAUTAMA PUTRADA 1, (Member, IEEE), IKKE DIAN OKTAVIANI
(Graduate Student Member, IEEE), MAMAN ABDUROHMAN 1, (Member,IEEE), MARIJN JANSSEN 2, HILAL HUDAN NUHA 1, (Senior Member, IEEE), AND SARWONO SUTIKNO 3 (Member, IEEE); February 2025

[10] "Secure Online Auction System": Pratiksha Nikalje 1, Pragati Randive 2, Shabana Machkuri3 Final Year Student, Department of Computer, Al Ameer College of Engineering, Pune, India, April 2025

[11] McAfee, R. Preston, and John McMillan. "Analyzing the Airwaves Auction." Journal of Economic Perspectives, vol. 10, no. 1, 2019, pp. 159

[12] Milgrom, Paul R., and Robert J. Weber. "Theory of Auctions and Competitive Bidding." Econometrica, vol. 50, no. 5, 2021, pp. 1089-1122.

[13] Noussair, Charles N., and Steven G. Medema. "The Impact of Auctions and Buyer Identity on Prices in Online Auctions: Results from a DCE Workshop.“ Journal of Economic Behavior & Organization, vol. 68, no. 2, 2018, pp. 436- 448.

[14] Plott, Charles R. "Auction Mechanisms: A Survey." Handbook of Experimental Economics Results, edited by Charles R. Plott and Vernon L. Smith, Elsevier, 2018.

[15] Plott, Charles R. "Vickrey Auctions in Theory and Practice." Economic Inquiry, vol. 17, no. 1, 2021, pp. 1-12

[16] Alberto Sonnino, Michał Krol, Argyrios G. Tasiopoulos, Ioannis Psaras, “AStERISK: Auction-based Shared Economy Resolution Markets for Blockchain Platform”, Workshop on Decentralized IoT Systems and Security (DISS) 2019 24 February 2019.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

ACKNOWLEDGEMENT

First, we must acknowledge the almighty God's choices and abundant blessings. His providence touched every piece of this project. He is the sole source of success because we are tools in the hands of God omnipotent. He guided, enlighten, gave energy and knowledge to make this project possible and successful.

We express our sincere gratitude to our beloved Principal Dr.R.KANNAN M.E., Ph.D., for his sincere endeavor in educating us in this premier institution.

Our sincere and heartfelt gratitude to Mr.V.RAGUNATH,M.E., Head of the Department for his kind help and guidance rendered during studies. We wholeheartedly acknowledge the words of inspiration given by our precious guide Mrs. R.ANBARASI,M.E., Assistant Professor for completing the project work.

We propose our sincere thanks to the project coordinator Mr.V.RAGUNATH,M.E., who has motivated to completed the project successfully.

We convey our sincere thanks to all the staff members and lab assistants of the Computer Science department. Last but not least, we would like to thank our parents and friends for lending us a helping hand in all endeavors during the project work and always providing us with the necessary motivation to complete this project successfully.