WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Bilateral Exchange System for Fresh Produce Selling using Auction Simulator

Mr. A. Akilan M.E. [*1], ***Mr. Ashiq Mohammed. J*** [*2], ***Mr. Gunal. M*** [*3], ***Mr. Vishnuprasath*** [*4]

[*1]*Head of The Department of computer science and engineering, MRK Institute of Technology, Kattumanarkoil, Tamil Nadu*
[*2,3,4] *UG Student of Computer science and engineering, MRK Institute of Technology, Kattumanarkoil, Tamil Nadu*

AB S T R A C T

This project addresses the challenges inherent in traditional fresh produce selling systems, which often rely on local markets and intermediaries, leading to limited reach and reduced profit margins for farmers. Maintaining consistent quality throughout the supply chain is crucial due to the perishable nature of fresh produce. To tackle these issues, this project proposes a Double Auction mechanism for produce trading systems. In this system, an e-commerce platform acts as the auctioneer, facilitating dynamic bidding between procurers and growers. The transparent and efficient nature of the double auction allows participants to adapt quickly to changing market dynamics, reducing the time of refrigeration and thereby lowering the carbon footprint of the agriculture supply chain. By providing access to market intelligence and enabling direct engagement between producers and procurers, the double auction system aims to improve efficiency and profitability while minimizing waste in the fresh produce industry.

Keywords: fresh produce, double auction mechanism, trading system, e-commerce platform, market intelligence, supply chain

1. INTRODUCTION

The trading of fresh produce, including fruits, vegetables, meats, and seafood, is essential to human sustenance and plays a pivotal role in our evolutionary history. However, traditional selling systems often face significant challenges, relying heavily on local markets and intermediaries, which limit market reach and reduce profit margins for producers. Moreover, maintaining consistent quality throughout the supply chain proves to be a daunting task in these systems, given the perishable nature of fresh produce and the need for quick turnover.

To address these challenges, this project introduces a novel approach: a Double Auction mechanism for fresh produce trading systems. This mechanism leverages an e-commerce platform to serve as the auctioneer, facilitating dynamic bidding between procurers and growers. By enabling direct engagement between producers and procurers, the Double Auction system promotes transparency, efficiency, and fair pricing. Additionally, it provides access to market intelligence, enabling participants to make informed decisions based on pricing trends, demand forecasts, and emerging consumer preferences.

Through this innovative approach, the project aims to revolutionize the fresh produce industry, reducing waste, lowering the carbon footprint of the agriculture supply chain, and ultimately improving profitability for producers while ensuring high-quality produce for consumers.

2. OBJECTIVES

- Develop a transparent and efficient trading platform for fresh produce.
- Facilitate direct engagement between producers and procurers.
- Provide access to market intelligence and pricing trends.
- Improve profitability for producers by reducing reliance on intermediaries.
- Minimize waste and lower the carbon footprint of the agriculture supply chain.
- Enhance overall efficiency and fairness in the fresh produce market.

3. METHODOLOGY

- Platform Development: Design and develop the e-commerce platform to serve as the auctioneer.

- Algorithm Design: Create a robust matching algorithm to pair procurers and growers based on their preferences, quantities, and bidding prices.
- User Registration: Implement a user registration system for buyers and sellers, collecting necessary information for participation.
- Auction Simulation: Develop a real-time auction simulator where buyers can place bids and sellers can list their produce.
- Transaction Handling: Design a system to handle transactions, including payment processing, delivery arrangements, and documentation.
- Feedback and Rating System: Establish a feedback and rating mechanism to maintain trust and accountability within the marketplace.

4. SYSTEM SPECIFICATION

4.1. HARDWARE REQUIREMENTS

- Dual-core processor (e.g., Intel Core i5 or AMD Ryzen 5 series).
- 8GB of RAM for handling concurrent user requests.
- 256GB SSD for faster data access and application performance.

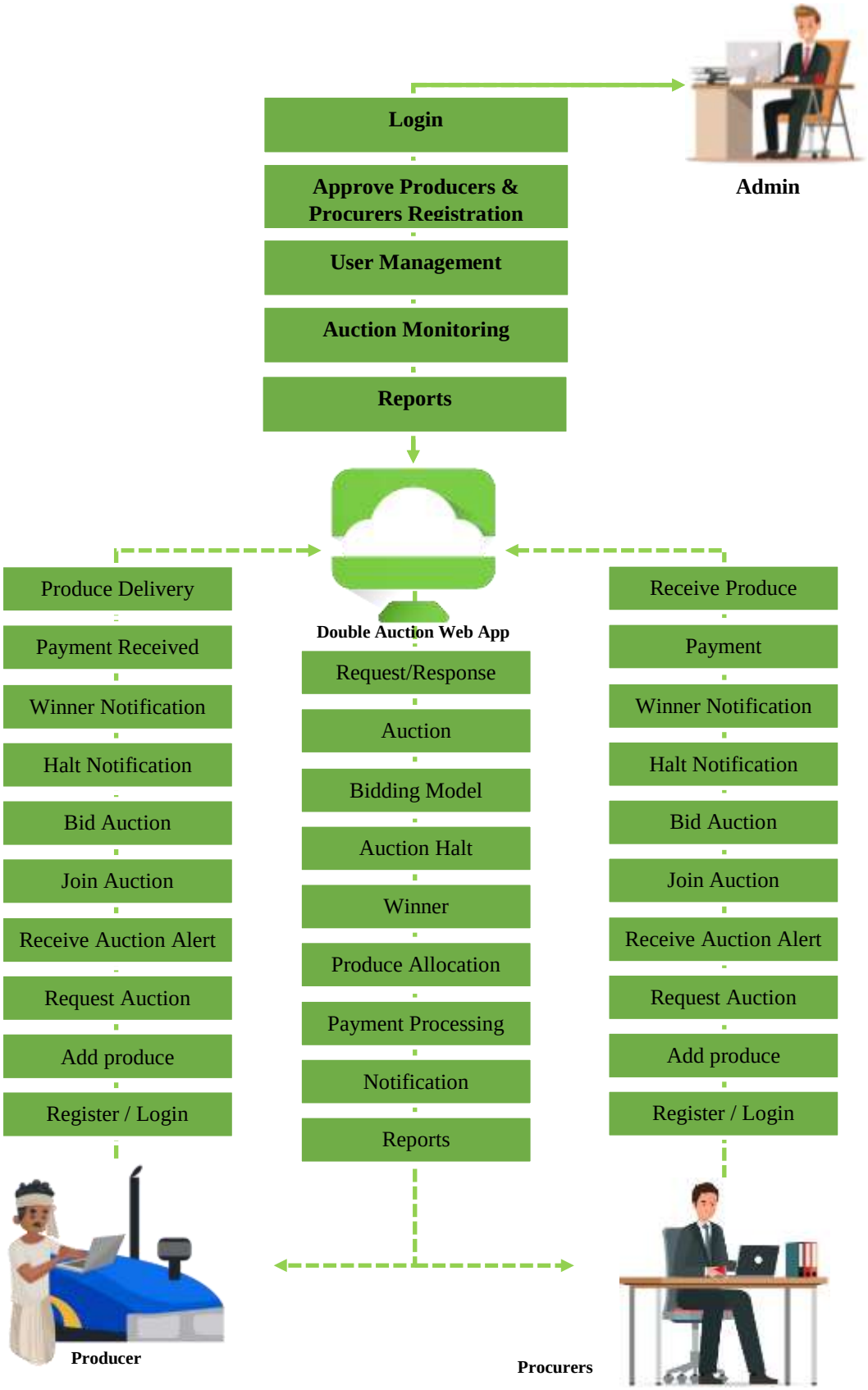
4.2. SOFTWARE REQUIREMENTS

- **Operating System:** Windows 10 or Windows 11 for the development environment.
- **Web Server:** Apache HTTP Server for hosting the web application.
- **Database Management System (DBMS):** MySQL 8.0 for Windows as the relational database management system.
- **Web Application Framework:** Flask 2.0 as the web framework for developing the application logic.
- **Programming Language:** Python 3.8 or later as the primary programming language for application development.
- **Front-End Technologies:**HTML5, CSS3, JavaScript for developing a responsive and user-friendly interface. Bootstrap 5 for streamlined and mobile-friendly design.
- **Data Science Libraries:** Pandas for data manipulation and analysis. NumPy for numerical operations and array handling. Scikit-learn for machine learning models and algorithms
- **Development and Deployment Tools:** Use an integrated development environment (IDE) such as PyCharm

5. SYSTEM ARCHITECTURE

5.1. SYSTEM ARCHITECTURE

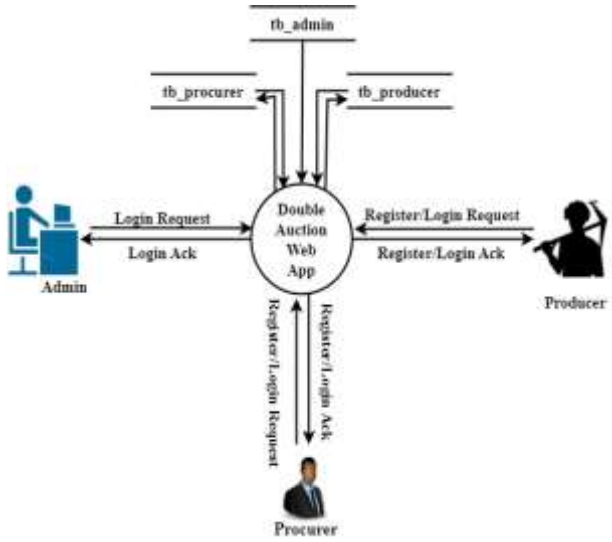
The system architecture for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator is depicted in the diagram. At its core is the Auction Simulator server, responsible for managing auctions, processing bids, and facilitating transactions. The server communicates with the Auction Interface, which serves as the frontend for both producers and procurers, allowing them to view auctions, place bids, and monitor results. The Producer Database stores information about growers and their produce, while the Procure Database holds data on procurers and their preferences. The system incorporates security features such as encryption and authentication to safeguard sensitive information and ensure the integrity of transactions. Overall, this architecture enables efficient and transparent trading, benefiting both producers and procurers in the fresh produce market.


Login
Approve Producers & Procurers Registration
User Management
Auction Monitoring
Reports
Admin
Double Auction Web App
Request/Response
Auction
Bidding Model
Auction Halt
Winner
Produce Allocation
Payment Processing
Notification
Reports
Produce Delivery
Payment Received
Winner Notification
Halt Notification
Bid Auction
Join Auction
Receive Auction Alert
Request Auction
Add produce
Register / Login
Producer
Receive Produce
Payment
Winner Notification
Halt Notification
Bid Auction
Join Auction
Receive Auction Alert
Request Auction
Add produce
Register / Login
Procurers

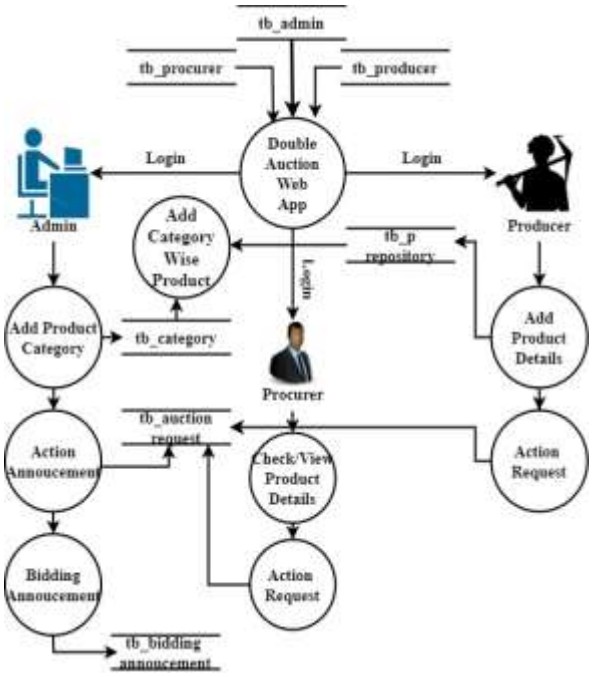
5.2 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator illustrates the flow of information within the system. At the center of the diagram is the Auction Simulator, which serves as the main processing unit. The diagram depicts the flow of data between various components, including producers, procurers, the auctioneer, and the auction interface. Producers submit information about their produce, including quantity and quality, to the system, which then organizes auctions based on this data. Procurers, on the other hand, submit bids for the produce they wish to purchase. The auctioneer oversees the auction process, confirming bids and announcing winners. Finally, the auction interface provides a user-friendly platform for producers and procurers to interact with the system. This DFD highlights the seamless flow of information within the system, facilitating efficient trading of fresh produce.

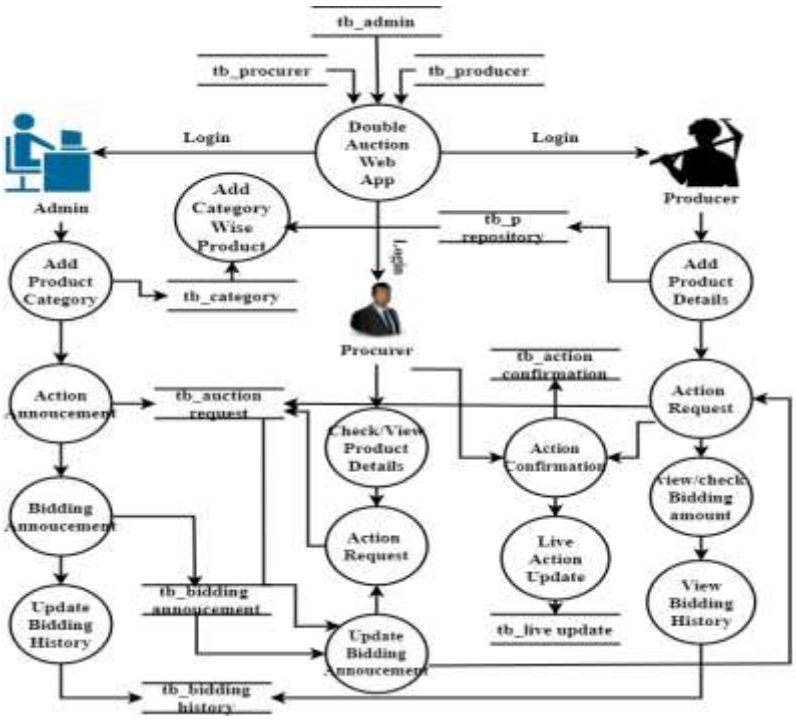
LEVEL 0



LEVEL 1



LEVEL 2



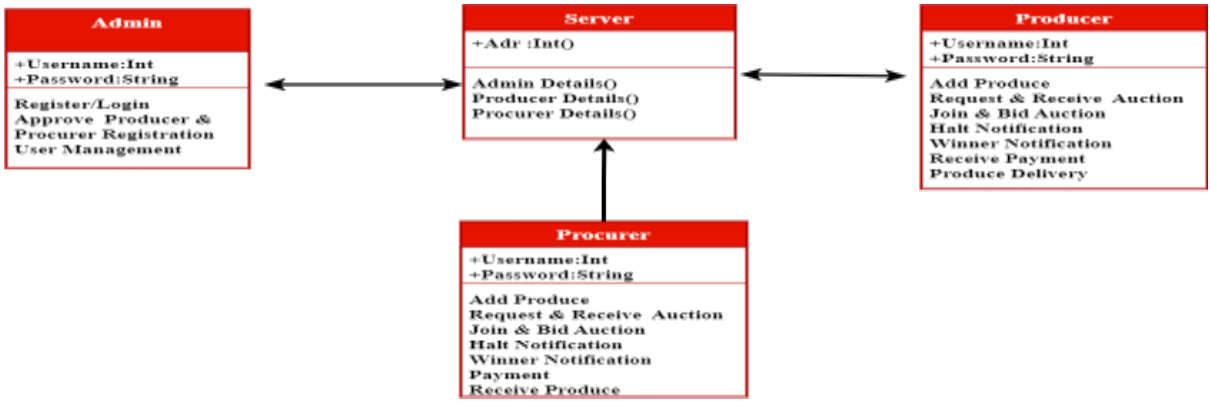
5.3. UML DIAGRAM

5.3.1. USE CASE DIAGRAM

The Use Case Diagram for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator outlines the various interactions between actors and the system. Actors include Producers, Procures, and the Auctioneer. Producers can register with the system, submit details about their produce, participate in auctions, and view auction results. Procures, on the other hand, can register, place bids on produce, and view auction outcomes. The Auctioneer oversees the entire auction process, including organizing auctions, confirming bids, and announcing winners. This diagram illustrates the key functionalities of the system and the roles of different actors in the produce trading process.

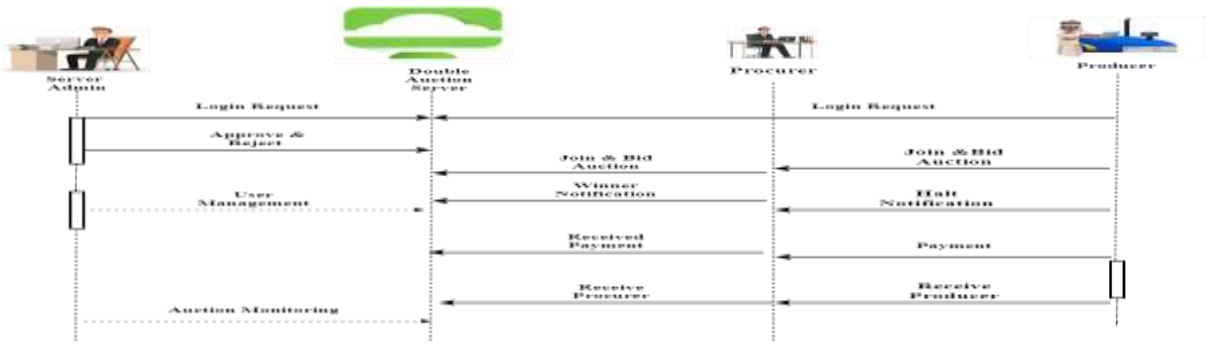
5.3.2. CLASS DIAGRAM

The Class Diagram for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator depicts the essential classes and their connections within the system. It outlines key entities such as Producers, Procures, and the Auctioneer, illustrating their roles and interactions. The diagram also represents vital components like Produce, Bids, and Auctions, detailing how they relate to each other. By providing a visual overview of the system's architecture, the Class Diagram aids in understanding the relationships between different elements and guides the development process effectively.



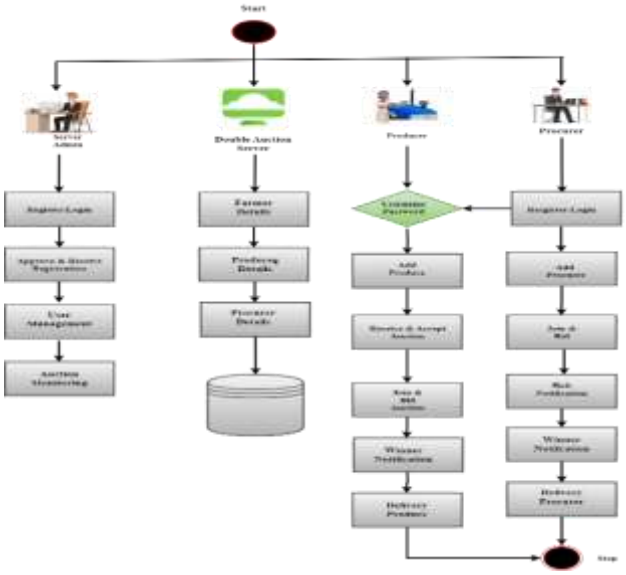
5.3.3. SEQUENCE DIAGRAM

The Sequence Diagram for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator illustrates the step-by-step interactions between various actors and components within the system. It outlines the flow of events during the auction process, starting from the submission of bids by Procures to the confirmation and allocation of produce by the Auctioneer. The diagram captures the dynamic nature of the auction mechanism, showcasing how bids are evaluated, prices are determined, and winners are selected. By visualizing these interactions, the Sequence Diagram helps stakeholders understand the sequence of actions required for successful produce trading and facilitates the implementation of the auction simulator with clarity and precision.



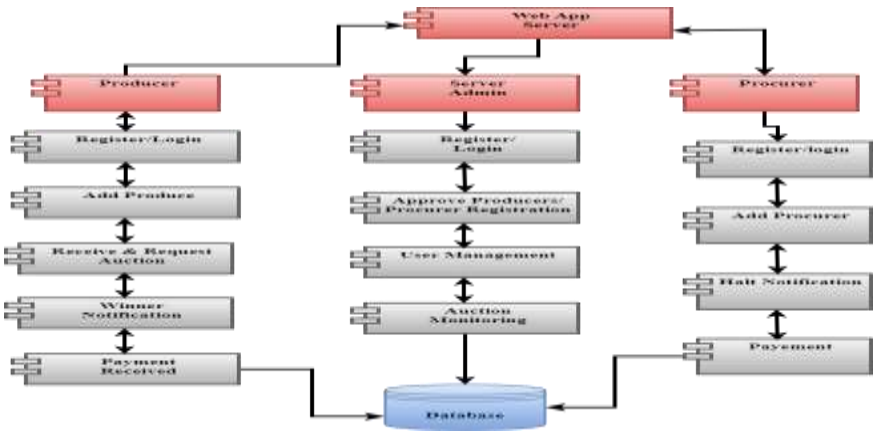
5.3.4. ACTIVITY DIAGRAM

The Activity Diagram offers a streamlined visual depiction of the auction process for fresh produce. It maps out key activities, like bid submission, evaluation, and winner selection, aiding in understanding the auction's flow. This diagram serves as a blueprint for implementing and managing the auction simulator efficiently.



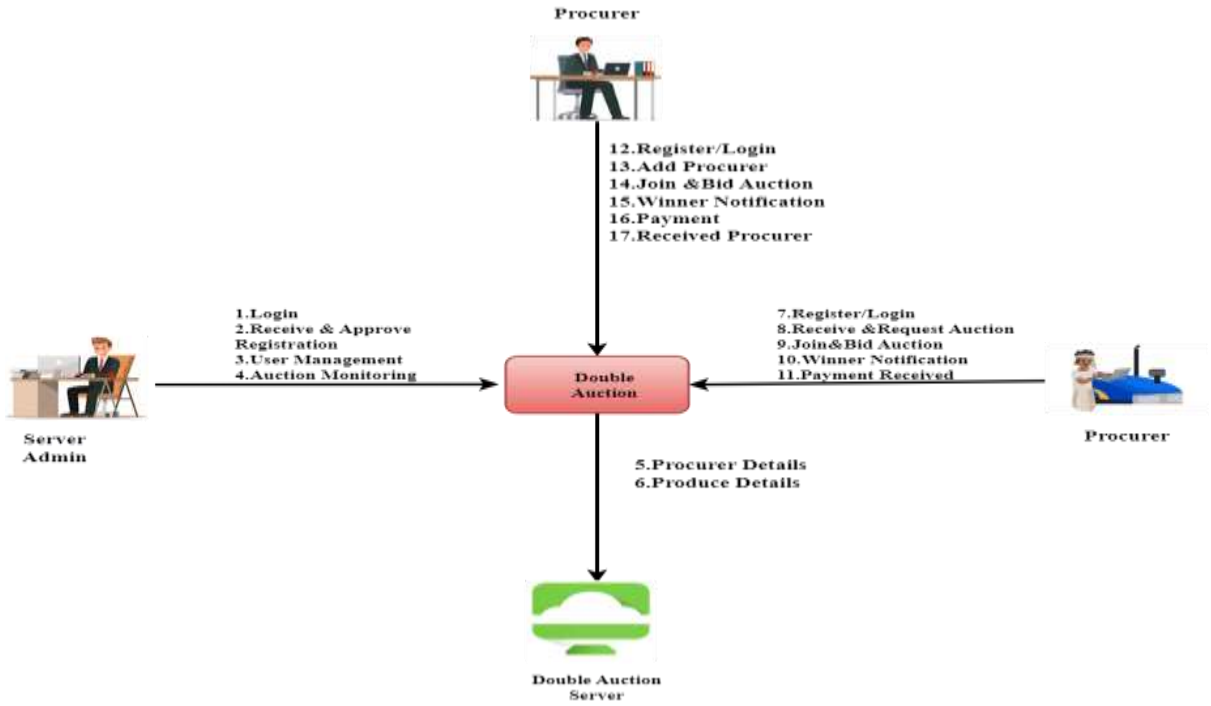
5.3.5. COMPONENET DIAGRAM

The Component Diagram outlines the modular structure of the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator. It identifies the key components such as the e-commerce platform, auction engine, user interface, database, and external APIs. Each component encapsulates specific functionalities, facilitating system scalability, maintainability, and interoperability. This diagram serves as a roadmap for system development, ensuring clear delineation of responsibilities and efficient integration of components.



5.3.6. COLLABORATION DIAGRAM

The Collaborate Diagram illustrates the interactions between components in the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator. It highlights how different modules and services communicate and collaborate to fulfill system functionalities. For instance, it demonstrates how the auction engine interacts with the database to retrieve auction data, how the user interface communicates with the auctioneer component to display auction results, and how external APIs are integrated to access market intelligence. This diagram provides a visual representation of the system's architecture, facilitating understanding of information flow and dependencies between components.

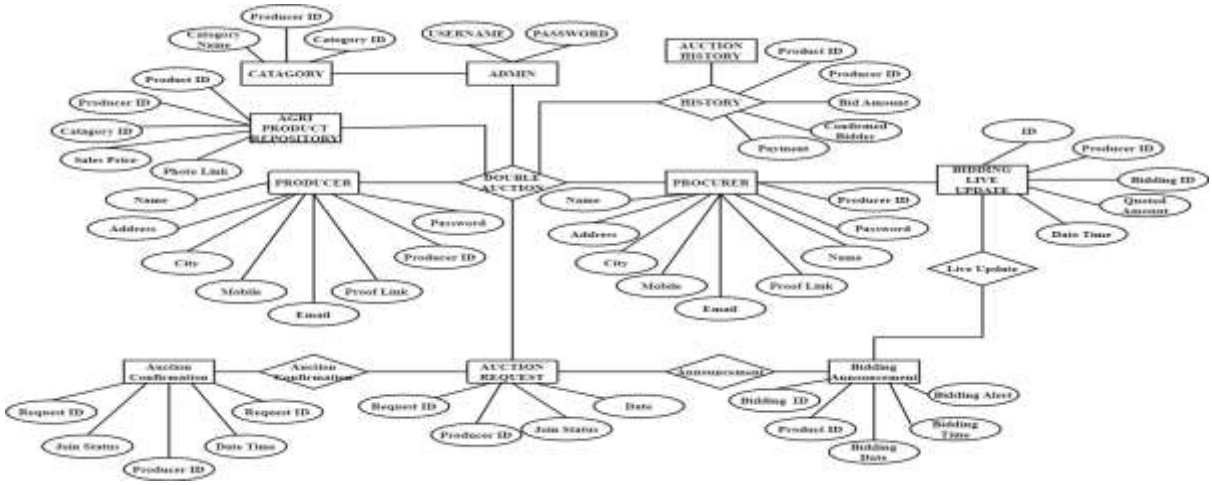


5.3.7. DEPLOYMENT DIAGRAM

The Deployment Diagram for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator depicts the physical deployment of system components across various nodes or hardware devices. It illustrates how the system is distributed across servers, databases, and other infrastructure elements. For example, it shows how the auction engine, database server, and web server are deployed on separate nodes, and how they interact with each other over the network. Additionally, it outlines the deployment of external services or APIs that the system relies on, such as market data providers or payment gateways. This diagram provides a clear visualization of the system's deployment architecture, aiding in understanding its scalability, redundancy, and performance characteristics.

5.4. ER DIAGRAM

The ER (Entity-Relationship) Diagram for the Bilateral Exchange System for Fresh Produce Selling using Auction Simulator provides a visual representation of the system's database schema and the relationships between different entities. In this diagram, entities such as Users, Posts, ChatMessages, and BlockedUsers are represented, along with their attributes and the associations between them. For instance, the Users entity includes attributes like UserID, username, email, and status, while the Posts entity includes attributes such as PostID, content, and post date. Relationships, such as the one-to-many relationship between Users and Posts (indicating that a user can create multiple posts), are also illustrated. This ER diagram serves as a blueprint for designing the system's database structure, ensuring data integrity and efficient storage and retrieval of information.



6. TEST CASE AND TEST REPORT

6.1 Test Cases

Producer End User Dashboard

1. **Test Case ID: PROD_REGISTRATION_001**
    - **Input:** Register as a new producer.
    - **Expected Result:** Producer account is successfully created.
    - **Actual Result:** Producer account is created with valid credentials.
    - **Status:** Pass
2. **Test Case ID: PROD_LOGIN_001**
    - **Input:** Log in as a registered producer.
    - **Expected Result:** Producer successfully logs into the dashboard.
    - **Actual Result:** Producer successfully accesses the dashboard with correct credentials.
    - **Status:** Pass
3. **Test Case ID: PROD_REQUEST_AUCTION_001**
    - **Input:** Request to initiate an auction.
    - **Expected Result:** Auction request is submitted successfully.
    - **Actual Result:** Auction request is successfully submitted.
    - **Status:** Pass
4. **Test Case ID: PROD_RECEIVE_AUCTION_INTIMATION_001**
    - **Input:** Await auction initiation notification.
    - **Expected Result:** Producer receives timely notification about the auction initiation.
    - **Actual Result:** Producer receives notification at the specified time.
    - **Status:** Pass
5. **Test Case ID: PROD_JOIN_AUCTION_001**
    - **Input:** Join the ongoing auction.
    - **Expected Result:** Producer successfully joins the auction.

- **Actual Result:** Producer is able to join the auction.
- **Status:** Pass

6. **Test Case ID: PROD_SUBMIT_BID_001**
    - **Input:** Submit a bid for the auction.
    - **Expected Result:** Producer's bid is successfully submitted.
    - **Actual Result:** Producer's bid is recorded accurately.
    - **Status:** Pass
7. **Test Case ID: PROD_RECEIVE_HALT_INTIMATION_001**
    - **Input:** Await halt notification for the auction.
    - **Expected Result:** Producer receives notification about the auction halt.
    - **Actual Result:** Producer receives halt notification as expected.
    - **Status:** Pass
8. **Test Case ID: PROD_RECEIVE_WINNER_NOTIFICATION_001**
    - **Input:** Await notification about auction winner determination.
    - **Expected Result:** Producer receives notification about auction winner determination.
    - **Actual Result:** Producer receives winner notification as expected.
    - **Status:** Pass
9. **Test Case ID: PROD_RECEIVE_PAYMENT_001**
    - **Input:** Await payment for the produce sold.
    - **Expected Result:** Producer receives payment for the sold produce.
    - **Actual Result:** Producer receives payment as expected.
    - **Status:** Pass
10. **Test Case ID: PROD_DELIVER_PRODUCT_001**
    - **Input:** Deliver the allocated produce to the winning procurers.
    - **Expected Result:** Producer successfully delivers the allocated produce.
    - **Actual Result:** Producer delivers produce to procurers as per allocation.
    - **Status:** Pass

Procurer End User Dashboard

1. **Test Case ID: PROC_REGISTRATION_001**
    - **Input:** Register as a new procurer.
    - **Expected Result:** Procurer account is successfully created.
    - **Actual Result:** Procurer account is created with valid credentials.
    - **Status:** Pass
2. **Test Case ID: PROC_LOGIN_001**
    - **Input:** Log in as a registered procurer.
    - **Expected Result:** Procurer successfully logs into the dashboard.
    - **Actual Result:** Procurer successfully accesses the dashboard with correct credentials.
    - **Status:** Pass
3. **Test Case ID: PROC_REQUEST_AUCTION_001**

- **Input:** Request to initiate an auction.
- **Expected Result:** Auction request is submitted successfully.
- **Actual Result:** Auction request is successfully submitted.
- **Status:** Pass

4. **Test Case ID: PROC_RECEIVE_AUCTION_INTIMATION_001**
    - **Input:** Await auction initiation notification.
    - **Expected Result:** Procurer receives timely notification about the auction initiation.
    - **Actual Result:** Procurer receives notification at the specified time.
    - **Status:** Pass
5. **Test Case ID: PROC_JOIN_AUCTION_001**
    - **Input:** Join the ongoing auction.
    - **Expected Result:** Procurer successfully joins the auction.
    - **Actual Result:** Procurer is able to join the auction.
    - **Status:** Pass
6. **Test Case ID: PROC_SUBMIT_BID_001**
    - **Input:** Submit a bid for the auction.
    - **Expected Result:** Procurer's bid is successfully submitted.
    - **Actual Result:** Procurer's bid is recorded accurately.
    - **Status:** Pass
7. **Test Case ID: PROC_RECEIVE_HALT_INTIMATION_001**
    - **Input:** Await halt notification for the auction.
    - **Expected Result:** Procurer receives notification about the auction halt.
    - **Actual Result:** Procurer receives halt notification as expected.
    - **Status:** Pass
8. **Test Case ID: PROC_RECEIVE_WINNER_NOTIFICATION_001**
    - **Input:** Await notification about auction winner determination.
    - **Expected Result:** Procurer receives notification about auction winner determination.
    - **Actual Result:** Procurer receives winner notification as expected.
    - **Status:** Pass
9. **Test Case ID: PROC_MAKE_PAYMENT_001**
    - **Input:** Make payment for the won produce.
    - **Expected Result:** Procurer successfully makes payment for the won produce.
    - **Actual Result:** Procurer's payment is processed successfully.
    - **Status:** Pass
10. **Test Case ID: PROC_RECEIVE_PRODUCT_001**
    - **Input:** Await delivery of the won produce.
    - **Expected Result:** Procurer receives the won produce as per allocation.
    - **Actual Result:** Procurer receives the produce as expected.
    - **Status:** Pass

6.2 TEST REPORT

Introduction

This test report aims to evaluate the functionality of the End User Dashboard for both producers and procurers in the "Bilateral Exchange System for Fresh Produce Selling using Auction Simulator" project.

Test Objective

The objective of this testing is to ensure that the End User Dashboard functions correctly, allowing producers and procurers to perform various tasks such as registration, login, auction participation, bid submission, payment processing, and product delivery.

Test Scope

The testing scope covers all functionalities of the End User Dashboard, including registration, login, auction participation, bid submission, payment processing, and product delivery, for both producers and procurers.

Test Environment

- Devices: Desktop and mobile devices.
- Browsers: Chrome, Firefox, Safari.
- Operating Systems: Windows, macOS, Linux.

Test Result

- TC001: User Registration with valid details - Pass
- TC002: Auction Organization with predefined rules - Pass
- TC003: Bid Submission by Producers and Procurers - Pass
- TC004: Bidding Confirmation - Pass
- TC005: Auction Halt at predefined conditions - Pass
- TC006: Winner Determination - Pass
- TC007: Produce Allocation - Pass
- TC008: Payment Processing - Pass
- TC009: Product Delivery - Pass
- TC010: Notification and Reporting - Pass

Test Conclusion

The Bilateral Exchange System has successfully passed all integration and functionality tests. The system demonstrates a seamless coordination of modules and exhibits the expected behavior of critical features.

7. CONCLUSION AND FUTURE SCOPE

CONCLUSION

In conclusion, the project proposed a transformative approach to the traditional methods of fresh produce trading. This project addresses key challenges faced by producers and procurers in existing systems, introducing a dynamic and efficient Double Auction Mechanism facilitated by an e-commerce platform as the auctioneer. By leveraging this innovative system, the project aims to overcome the limitations of traditional local markets and direct sales, providing a broader market reach beyond geographic boundaries. The introduction of real-time bidding, bid confirmation processes, and dynamic price announcements ensures a transparent and streamlined auction experience for all participants. The integration of market intelligence into the platform empowers stakeholders with essential information on pricing trends, demand forecasts, and emerging consumer preferences. This knowledge facilitates informed decision-making, contributing to a more adaptive and responsive fresh produce market. The project also addresses issues such as middlemen dependency, limited shelf life, and quality consistency throughout the supply chain. Through the efficient allocation of produce and swift payment processes, it aims to improve profit margins for growers while minimizing losses associated with spoilage and delays. Furthermore, the system's adaptability to the perishable and fluctuating nature of fresh produce, along with its reduced refrigeration time, aligns with sustainability goals, contributing to a more environmentally conscious agricultural supply chain. In essence, the project is poised to revolutionize the fresh produce market, offering a dynamic, transparent, and efficient platform that benefits both producers and procurers. The project represents a significant step towards a sustainable and profitable future for the fresh produce industry.

FUTURE ENHANCEMENT

The project lays a strong groundwork for modernizing fresh produce trading. To enhance its effectiveness and reach, future enhancements could include blockchain integration for enhanced transparency, smart contracts for streamlined processes, machine learning for predictive analytics, mobile applications for improved accessibility, and IoT sensors for real-time quality monitoring. These advancements promise to further elevate the system's capabilities, ensuring a more efficient, transparent, and accessible marketplace for growers and procurers alike.

References

1. Smith, Vernon L. "An Experimental Study of Competitive Market Behavior." Journal of Political Economy, vol. 70, no. 2, 1962, pp. 111-137.
2. Kagel, John H., and Dan Levin. "Auctions: A Survey of Experimental Research." Handbook of Experimental Economics, edited by John H. Kagel and Alvin E. Roth, Princeton University Press, 1995.
3. Shogren, Jason F., et al. "Auction Mechanisms and the Measurement of WTP and WTA." Resource and Energy Economics, vol. 22, no. 3, 2000, pp. 255-269.
4. Kagel, John H., and Dan Levin. "Behavior in Multi-Unit Demand Auctions: Experiments with Uniform Price and Dynamic Vickrey Auctions." Econometrica, vol. 59, no. 6, 1991, pp. 1487-1518.
5. 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, 2008, pp. 436-448.
6. Bulow, Jeremy, and Paul Klemperer. "Auctions vs. Negotiations." American Economic Review, vol. 86, no. 1, 1996, pp. 180-194.
7. Cramton, Peter, and Lawrence M. Ausubel. "The Clock-Proxy Auction: A Practical Combinatorial Auction Design." Games and Economic Behavior, vol. 52, no. 2, 2005, pp. 329-354.
8. Cox, James C., et al. "Bidding Behavior in First-Price Sealed-Bid Auctions: Use of Computerized Nash Competitors." International Journal of Game Theory, vol. 21, no. 3, 1992, pp. 279-295.
9. McAfee, R. Preston, and John McMillan. "Analyzing the Airwaves Auction." Journal of Economic Perspectives, vol. 10, no. 1, 1996, pp. 159-175.
10. Engelbrecht-Wiggans, Richard. "Auctions and Bidding Models: A Survey." Journal of Economic Surveys, vol. 3, no. 2, 1989, pp. 95-121.
11. Gneezy, Uri, and Aldo Rustichini. "A Fine Is a Price." Journal of Legal Studies, vol. 29, no. 1, 2000, pp. 1-17.
12. Plott, Charles R. "Auction Mechanisms: A Survey." Handbook of Experimental Economics Results, edited by Charles R. Plott and Vernon L. Smith, Elsevier, 2008.
13. Wilson, Robert B. "Auctions of Shares." Quarterly Journal of Economics, vol. 82, no. 1, 1968, pp. 153-179.
14. Plott, Charles R. "Vickrey Auctions in Theory and Practice." Economic Inquiry, vol. 17, no. 1, 1979, pp. 1-12.
15. Easley, David, and Maureen O'Hara. "Liquidity and Valuation in an Uncertain Market." Journal of Financial Economics, vol. 19, no. 2, 1987, pp. 351-360.
16. Kagel, John H., and Dan Levin. "Information Impact and Allocation Rules in Auctions with Affiliated Private Values: A Laboratory Study." Econometrica, vol. 55, no. 6, 1987, pp. 1275-1304.
17. Milgrom, Paul R., and Robert J. Weber. "A Theory of Auctions and Competitive Bidding." Econometrica, vol. 50, no. 5, 1982, pp. 1089-1122.
18. Klemperer, Paul. "Auction Theory: A Guide to the Literature." Journal of Economic Surveys, vol. 13, no. 3, 1999, pp. 227-286.
19. Roth, Alvin E. "The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics." Econometrica, vol. 70, no. 4, 2002, pp. 1341-1378.
20. Ashenfelter, Orley, and Kathryn Graddy. "Auctions and the Price of Art." Journal of Economic Literature, vol. 41, no. 3, 2003, pp. 763-787.

APPENDIX

SAMPLE CODING

Packages

```
from flask import Flask, render_template, Response, redirect, request, session, abort, url_for

import mysql.connector

import hashlib
```

```
import datetime
import calendar
import random
import csv
from random import randint
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from PIL import Image
import urllib.request
import urllib.parse
Database Connection
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
charset="utf8",
database="double_auction"
Login
def login():
msg=''
if request.method=='POST':
uname=request.form['uname']
pwd=request.form['pass']
cursor = mydb.cursor()
cursor.execute('SELECT * FROM da_producer WHERE producer_id = %s AND password = %s AND approved_status=1', (uname, pwd))
account = cursor.fetchone()
if account:
session['username'] = uname
return redirect(url_for('prd_home'))
else:
msg = 'Incorrect username/password! or access not provided'
Producer Register
def reg_prd():
msg=""
filename=""
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM da_producer")
maxid = mycursor.fetchone()[0]
```

```
if maxid is None:
maxid=1
producer_id="PD"+str(maxid)
if request.method=='POST':
name=request.form['name']
address=request.form['address']
city=request.form['city']
mobile=request.form['mobile']
email=request.form['email']
producer_id=request.form['producer_id']
pass1=request.form['pass']
now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
mycursor.execute("SELECT count(*) from da_producer where producer_id=%s",(producer_id,))
cnt = mycursor.fetchone()[0]
if cnt==0:
file = request.files['proof_link']
if file.filename == '':
flash('No selected file')
return redirect(request.url)
if file:
fn=file.filename
filename="P"+str(maxid)+fn
#fn1 = secure_filename(fn)
file.save(os.path.join("static/upload", filename))
sql = "INSERT INTO da_producer(id,name,address,city,mobile,email,proof_link,producer_id,password,approved_status,register_date) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,name,address,city,mobile,email,filename,producer_id,pass1,'0',rdate)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
```

Procurer Request

```
def prc_addproduct():
msg=""
act = request.args.get("act")
fnn=""
mdata=[]
uname=""
if 'username' in session:
```

```
uname = session['username']
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM da_procurer where procurer_id=%s",(uname,))
data = mycursor.fetchone()
mycursor.execute("SELECT * FROM da_category")
data1 = mycursor.fetchall()
if request.method=='POST':
category=request.form['category']
product=request.form['product']
qty=request.form['qty']
details=request.form['details']
mycursor.execute("SELECT max(id)+1 FROM da_product")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO da_product(id,procurer_id,category,product,quantity,description,pro_type) VALUES (%s, %s, %s, %s, %s, %s,%s)"
val = (maxid,uname,category,product,qty,details,'buy')
mycursor.execute(sql, val)
mydb.commit()
mss="Procurer ID:"+uname+", Product: "+product+", Quantity: "+qty
```

Auction Request

```
def prc_home():
msg=""
st=""
uname=""
if 'username' in session:
uname = session['username']
act=request.args.get("act")
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM da_procurer where procurer_id=%s",(uname,))
data = mycursor.fetchone()
pcid=uname
name=data[1]
mycursor.execute("SELECT * FROM da_product where status=0 && pro_type='sale'")
data2= mycursor.fetchall()
if act=="join":
pid=request.args.get("pid")
mycursor.execute("SELECT * FROM da_product where id=%s",(pid,))
dat1= mycursor.fetchone()
```

```
producer_id=dat1[1]
mycursor.execute("SELECT count(*) FROM da_auction_request where procurer_id=%s && product_id=%s && pro_type='sale'",(uname,pid))
cnt= mycursor.fetchone()[0]
if cnt==0:
mycursor.execute("SELECT max(id)+1 FROM da_auction_request")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO da_auction_request(id,procurer_id,producer_id,product_id,pro_type) VALUES (%s, %s, %s, %s,%s)"
val = (maxid,uname,producer_id,pid,'sale')
mycursor.execute(sql, val)
mydb.commit()
msg="ok"
else:
msg="fail"
```

Auction

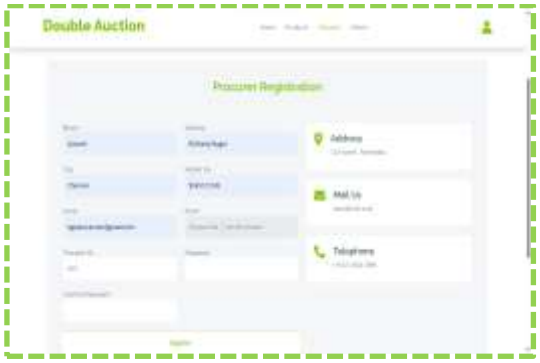
```
def auction():
msg=""
st=""
act = request.args.get("act")
pcid = request.args.get("pcid")
name = request.args.get("name")
bidmsg=""
fnn=""
uname=""
bdata=[]
lbid1=0
if 'username' in session:
uname = session['username']
ff2=open("static/bid.txt","r")
value=ff2.read()
ff2.close()
if value=="":
s=1
else:
ff.close()
v3=ms.split("|")
tot=len(v3)
```

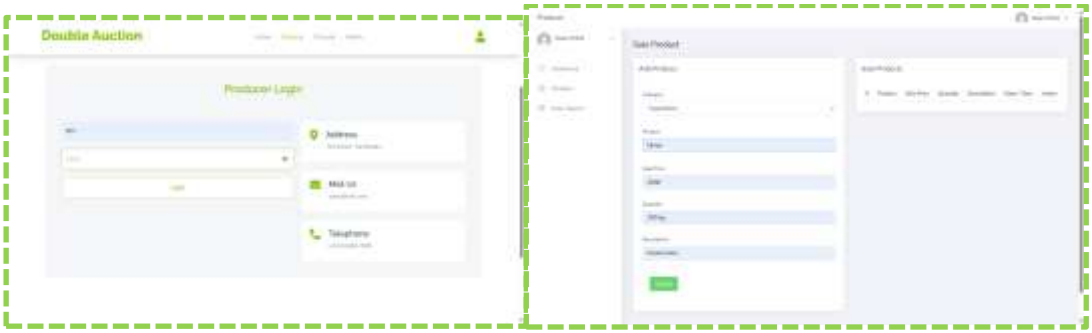
```
if tot>=1:
ff=open("static/lastbid.txt","r")
lbid=ff.read()
ff.close()
###
for v4 in v3:
dtt=[]
v5=v4.split(",")
if lbid1>17:
msg="done"
elif lbid1>17:
bidmsg="Last Auction Rs."+str(value2)+", by "+bu+", Auction Halted"
elif lbid1>15:
bidmsg="Last Auction Rs."+str(value2)+", by "+bu+", Auction will be closed...3"
elif lbid1>12:
bidmsg="Last Auction Rs."+str(value2)+", by "+bu+", Auction will be closed...2"
elif lbid1>9:
bidmsg="Last Auction Rs."+str(value2)+", by "+bu+", Auction will be closed...1"
elif lbid1>7:
bidmsg="Last Auction Rs."+str(value2)+", by "+bu
Auction Report
def auction_msg():
msg=""
act = request.args.get("act")
pcid = request.args.get("pcid")
name = request.args.get("name")
bidmsg=""
fnn=""
uname=""
bdata=[]
lbid1=0
if 'username' in session:
uname = session['username']
ff=open("static/bid.txt","r")
ms=ff.read()
ff.close()
v3=ms.split("|")
tot=len(v3)
if tot>=1:
```

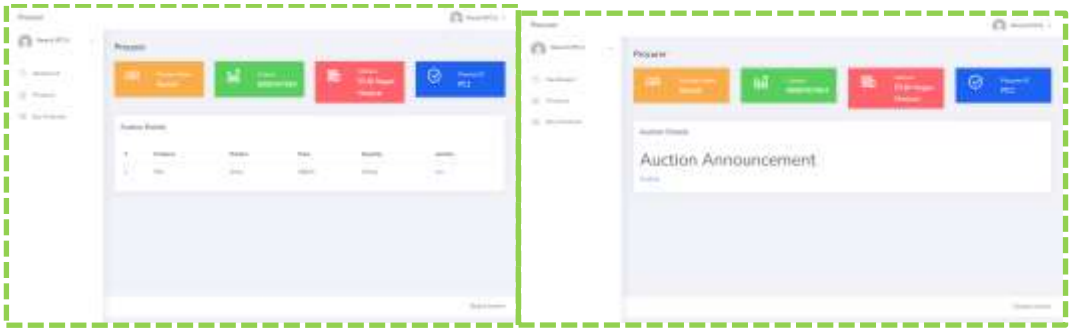
```
ff=open("static/lastbid.txt","r")
lbid=ff.read()
ff.close()
lbid1=int(lbid)+1
lbid2=str(lbid1)
ff=open("static/lastbid.txt","w")
ff.write(lbid2)
ff.close()
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM da_producer where producer_id=%s",(bu,))
data = mycursor.fetchone()
email=data[5]
if pcid==bu:
st="1"
mess="Dear "+name+", Producer ID"+bu+" You have won the Auction, Amount to Sale for Rs."+value2
mycursor.execute("update da_product set status=2,producer_id=%s,bid_amount=%s where id=%s",(bu,value2,pid))
mydb.commit()
for v4 in v3:
dtt=[]
v5=v4.split(",")
dtt.append(v5[0])
dtt.append(v5[1])
dtt.append(v5[2])
dtt.append(v5[3])
mycursor.execute("SELECT max(id)+1 FROM da_report2")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO da_report2(id,procurer_id,product_id,producer_id,amount,bdate,btime) VALUES (%s, %s, %s, %s,%s,%s,%s)"
val = (maxid,pcid,pid,v5[0],v5[1],v5[2],v5[3])
mycursor.execute(sql, val)
mydb.commit()
Payment
def pay():
msg=""
st=""
pid = request.args.get("pid")
act = request.args.get("act")
fnn=""
```

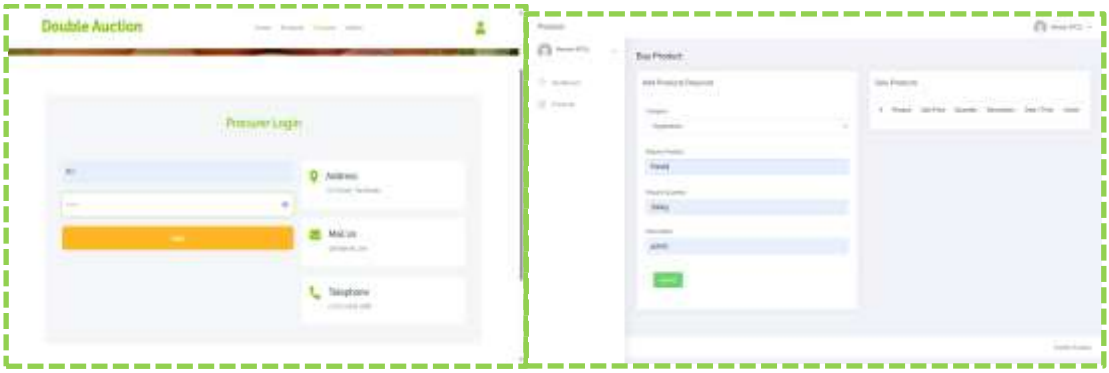
```
uname=""
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM da_procurer where procurer_id=%s",(uname,))
data = mycursor.fetchone()
pcid=uname
name=data[1]
if request.method=='POST':
card=request.form['card']
mycursor.execute("update da_product set pay_st=1 where id=%s",(pid,))
mydb.commit()
msg="success"
```

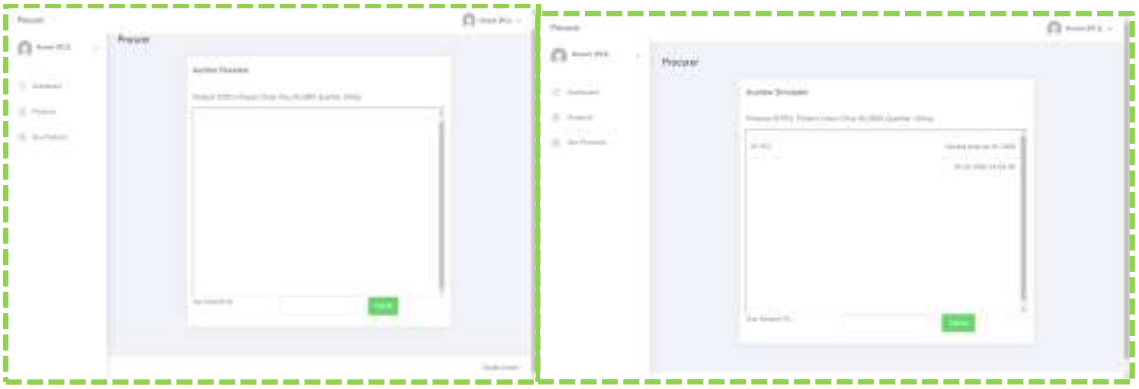
SCREEN SHOT

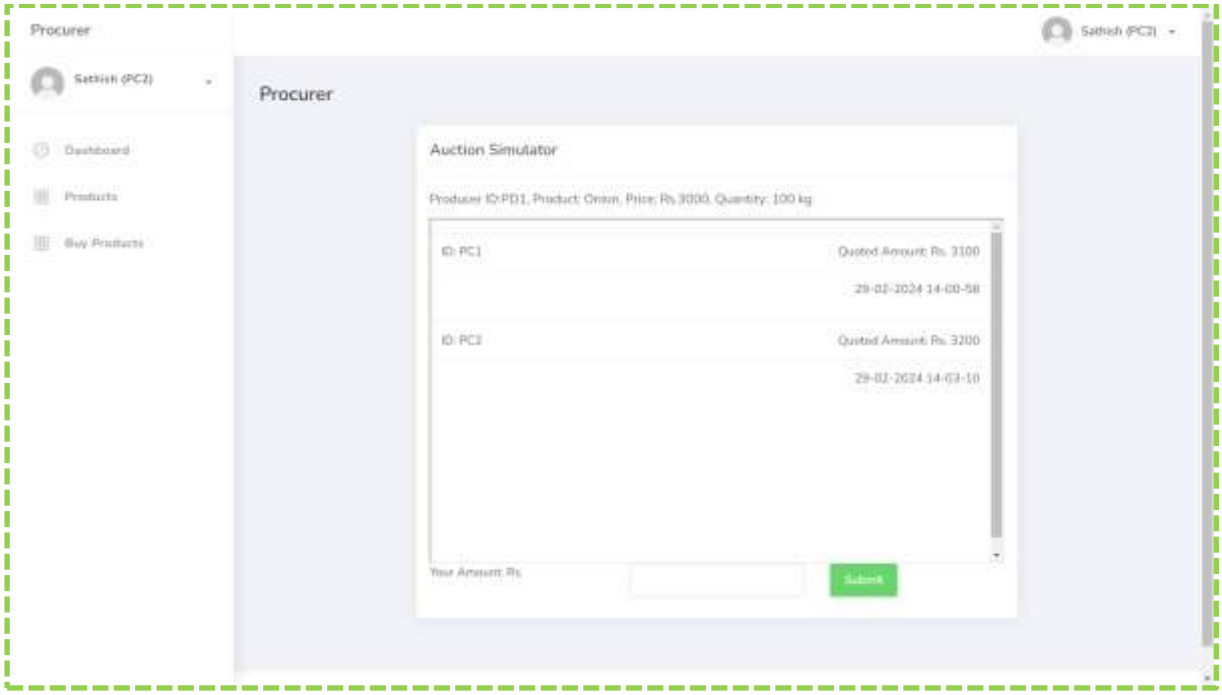


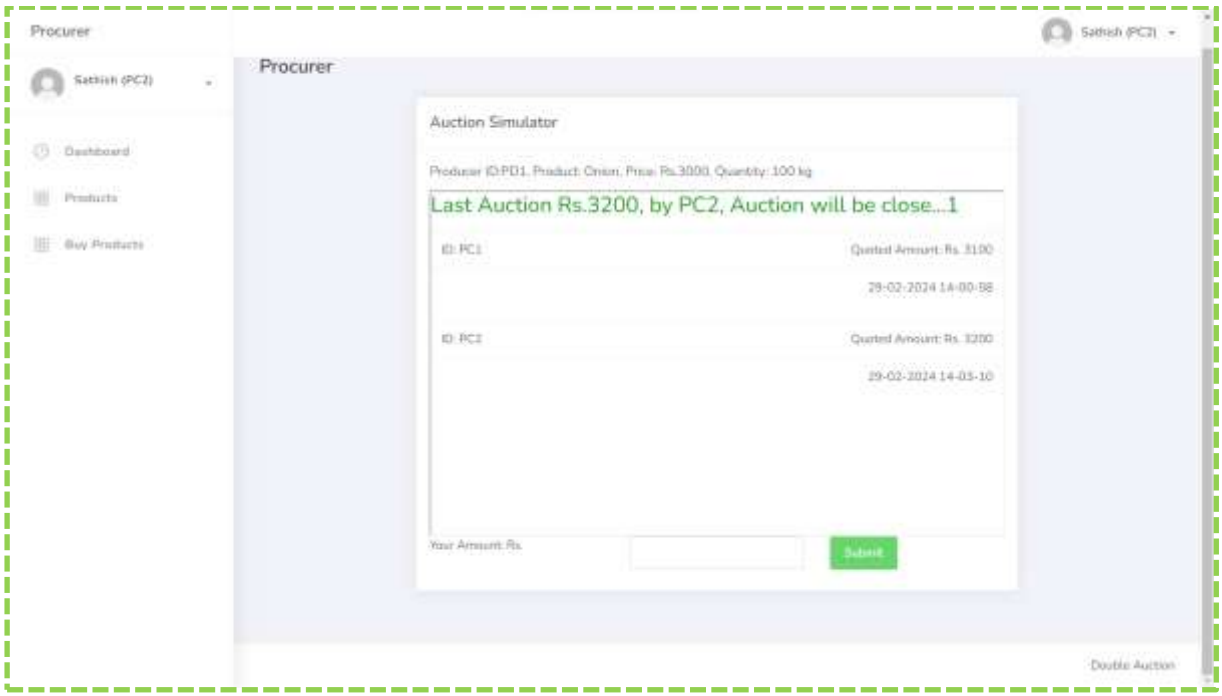

Double Auction
Producer Login

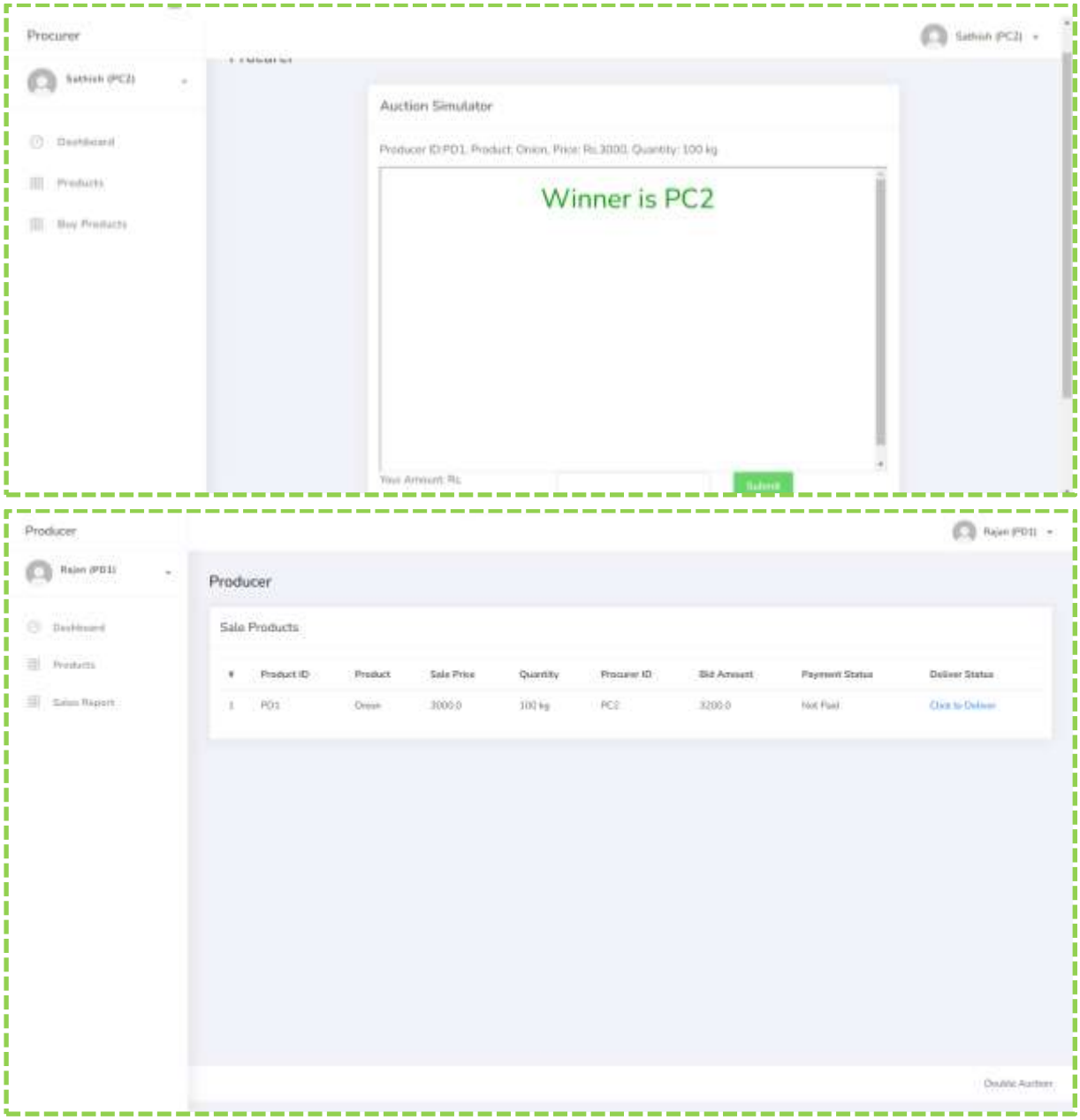

Auction Announcement

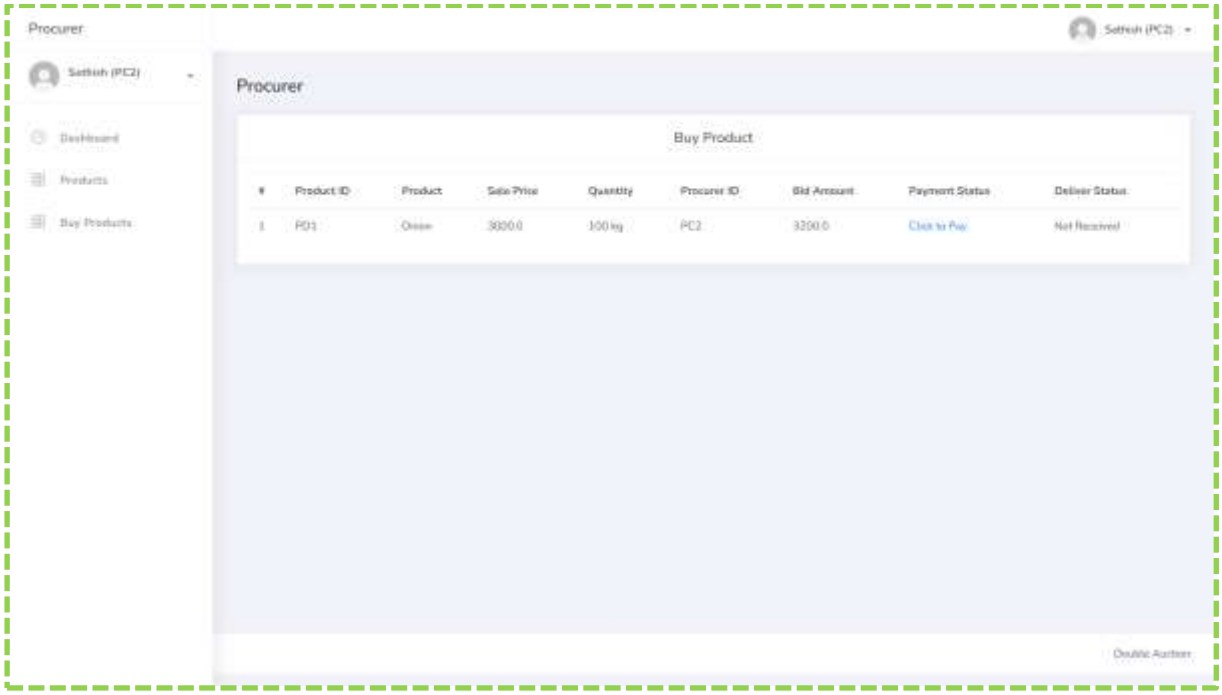

Double Auction
Procurer Login

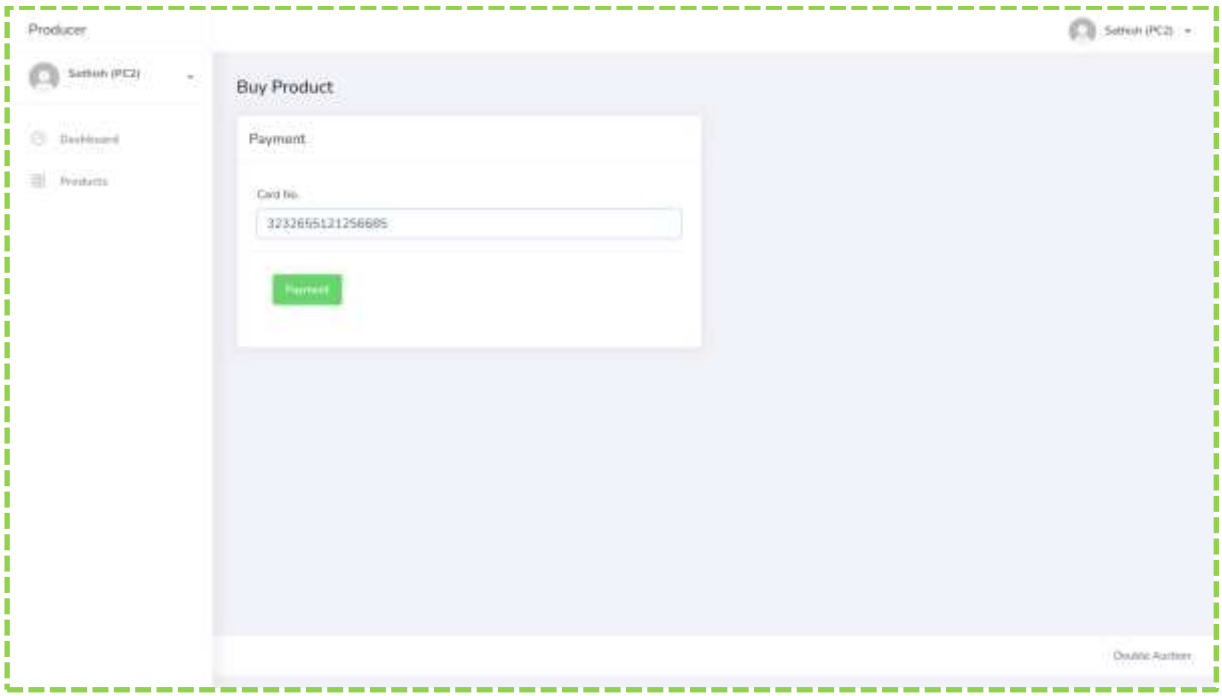



Procurer
Sathish (PC2)
Dashboard
Products
Buy Products
Procurer
Auction Simulator
Producer ID:PD1, Product: Onion, Price: Rs.3000, Quantity: 100 kg
ID: PC1
Quoted Amount: Rs. 3100
29-02-2024 14-00-58
ID: PC2
Quoted Amount: Rs. 3200
29-02-2024 14-03-10
Your Amount: Rs.
Submit


Procurer
Sathish (PC2)
Dashboard
Products
Buy Products
Procurer
Auction Simulator
Producer ID:PD1, Product: Onion, Price: Rs.3000, Quantity: 100 kg
Last Auction Rs.3200, by PC2, Auction will be close...1
ID: PC1
Quoted Amount: Rs. 3100
29-02-2024 14-00-58
ID: PC2
Quoted Amount: Rs. 3200
29-02-2024 14-03-10
Your Amount: Rs.
Submit
Double Auction


Procurer
Auction Simulator
Winner is PC2
Producer
Producer
Sale Products


Procurer
Procurer
Buy Product
Product ID
Product
Sale Price
Quantity
Procurer ID
Bid Amount
Payment Status
Deliver Status


Producer
Buy Product
Payment
Card No.
3232655121256685
Payment