

AUCTION MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

S.N.MOHANAA(2303811710422098)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621112

BONAFIDE CERTIFICATE

Certified that this project report on “**AUCTION MANAGEMENT SYSTEM**” is the bonafide work of **S.N.MOHANAA(2303811710422098)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E., (Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E., (Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on.....3/12/2024.....

CGB1201-JAVA PROGRAMMING
Mr.MANJARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER


CGB1201-JAVA PROGRAMMING
Mrs.AARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**AUCTION MANAGEMENT SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



S.N.MOHANAA

Place: Samayapuram

Date: 03/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The provided Java-based auction system demonstrates a basic yet functional model for managing and participating in auctions using AWT (Abstract Window Toolkit) for graphical user interface (GUI) development. The application is structured into several modules, including user selection, login, auction management, participant interaction, and bid handling. The system begins with a user selection screen where the user can choose their role as either an "Auctioneer" or a "Participant." Upon successful login, Auctioneers are presented with a screen that allows them to add auction items, set a base price, and start the auction by specifying a time limit. A countdown timer is used to track the remaining time of the auction, and once the auction concludes, the system determines and displays the highest bidder as the winner. Participants, on the other hand, can view the ongoing auction's details and place bids by entering their name and the bid amount. The system ensures that bids can only be placed while the auction is active. All bids are collected in an Array List, and the Auctioneer has access to the current list of bids and can update the status in real-time. The code demonstrates fundamental concepts of event-driven programming, handling user input, and managing dynamic UI updates based on user interaction.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>The Java-based auction system provides a simple graphical interface for managing and participating in auctions using AWT. It consists of separate modules for user selection, login, auction management, and bid placement. Auctioneers can add items, set prices, and start timed auctions, while participants can place bids during the auction. The system tracks bids and determines the winner based on the highest bid at the end of the auction. The application demonstrates core Java concepts such as event-driven programming, user interaction, and dynamic updates to the interface. While basic, it offers a foundation for further enhancements like data validation, persistent storage, and network support for remote participation.</p>	<p>PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3</p>	<p>PSO1 -3 PSO2 -3 PSO3 -3</p>

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
2	PROJECT METHODOLOGY	
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	MODULE DESCRIPTION	
	3.1 User Selection Module	4
	3.2 Login Module	4
	3.3 Auctioneer Module	4
	3.4 Participant Module	4
	3.5 Bid Management Module	4
	3.6 Timer Module	
	3.7 Participant Details Module	
	3.8 Winner Declaration Module	
4	CONCLUSION & FUTURE SCOPE	
	4.1 Conclusion	6
	4.2 Future Scope	6
	REFERENCES	7
	APPENDIX A (SOURCE CODE)	8
	APPENDIX B (SCREENSHOTS)	17

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Auction Management System is to provide a simple and interactive platform for conducting auctions, allowing auctioneers to manage items, set auction parameters, and monitor bids, while enabling participants to place bids in real time. The system aims to streamline the auction process by automating bid tracking, timing, and winner declaration within a user-friendly interface.

1.2 Overview

The Auction Management System is a graphical application developed using Java AWT that facilitates the process of conducting auctions. It provides distinct interfaces for auctioneers and participants, starting with a user selection screen followed by a login system. Auctioneers can add items, set auction details, and manage bids with a timer, while participants can view auction details and place bids. The system tracks all bids, determines the highest bidder at the end of the auction, and announces the winner, creating an interactive and efficient auction platform.

1.3 Java Programming Concepts

1. Object-Oriented Programming (OOP):

- Classes and objects (Auctioneer Screen, Participant Screen, Bid, etc.).
- Encapsulation (use of private fields with public getters and setters).

2. Event Handling:

- Action listeners (ActionListener) to respond to button clicks.
- Window listeners (Window Adapter) to handle window closing events.
- Dynamic UI updates based on user interactions.

3. Collections Framework:

- Use of Array List to manage and store bids.

4. Exception Handling:

- Handling invalid input with try-catch blocks (e.g., parsing bid amounts).

5. Timers:

- Use of `javax.swing.Timer` to implement the countdown timer for the auction.

6. String Handling:

- String comparisons and manipulations (e.g., validating user credentials, displaying bid details).

7. Control Flow:

- Conditional statements (if-else) for decision-making.
- Loops (implicit in `stream().max()` for determining the highest bid).

These concepts collectively enable the implementation of a functional and interactive auction system.

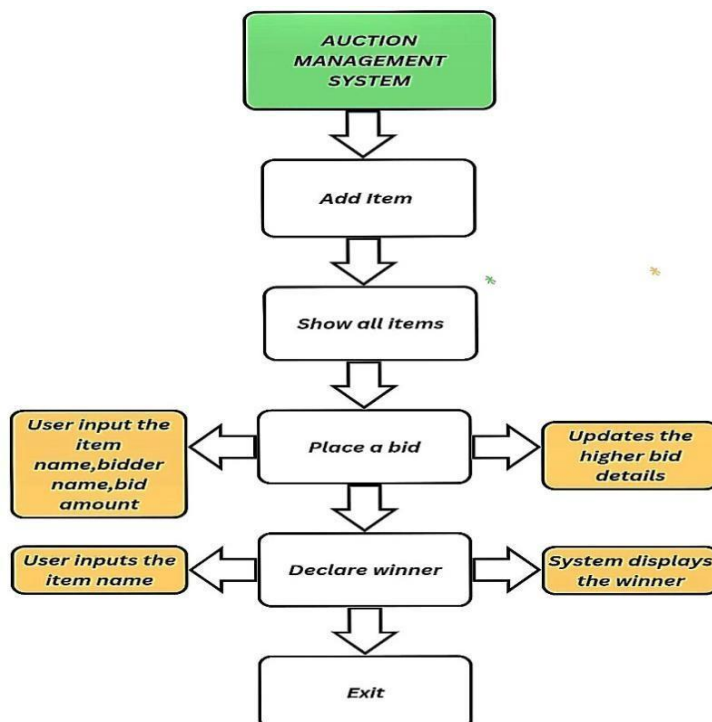
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the given auction system code focuses on enhancing its functionality, usability, and scalability. This includes improving data validation to prevent empty or invalid inputs, ensuring thread safety for concurrent access to shared resources, and upgrading the user interface by transitioning from AWT to a modern library like Swing or JavaFX. Additionally, the system can be extended with persistent storage to save auction data and a client-server architecture to allow remote participants to join auctions. These improvements aim to create a more robust, user-friendly, and scalable auction management application.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1 User Selection Module:

Displays a screen for users to select their role (Auctioneer or Participant).

3.2 Module 2 Login Module:

- Handles user authentication for both Auctioneers and Participants.
- Ensures secure access to the auction system.

3.3 Module 3 Auctioneer Module:

Enables auctioneers to set up auction items, specify base prices, and manage auction durations. Tracks bids, declares the highest bidder, and provides buttons for auction management, including starting auctions and viewing participants.

3.4 Module 4 Participant Module:

- Allows participants to view auction details and place bids during an active auction.
- Updates participants on the auction's status and their bid outcomes.

3.5 Module 5 Bid Management Module:

- Stores and manages all bids submitted during the auction.
- Provides methods to add bids, retrieve bid details, and determine the highest bid.

3.6 Module 6 Timer Module:

Implements a countdown timer to manage the auction's time limit. Automatically ends the auction and triggers winner declaration when the timer expires.

3.7 Module 7 Participant Details Module:

A newly added module to display a detailed list of participants and their bids. Provides a separate screen for auctioneers to review bid information transparently.

3.8 Module 8 Winner Declaration Module:

- Determines the highest bidder once the auction concludes.
- Announces the winner and displays bid details on the auction screen.

CHAPTER 5

CONCLUSION & FUTURE SCOPE

CONCLUSION

In conclusion, the provided auction system code serves as a foundational framework for a basic, interactive auction application using Java's AWT. It effectively manages the interaction between Auctioneers and Participants through dedicated screens for user selection, login, and role-based functionality. The system allows Auctioneers to set up and manage auctions, track bids, and declare winners, while Participants can view auction details and place bids in real-time. The modular design ensures clear separation of concerns, making it easy to expand and improve. While the current implementation offers basic functionality, it can be enhanced with better user interface design, data validation, persistent storage, and network support for scalability, transforming it into a fully functional online auction system.

FUTURE SCOPE

The future scope of this auction system lies in enhancing its functionality, scalability, and user experience. Transitioning from AWT to more modern frameworks like Swing or JavaFX can provide a richer and more intuitive interface. Adding real-time bid updates using threads or networking capabilities would enable remote participants to join the auction. Integrating a database for persistent storage of auction items, bid histories, and user credentials would make the system more robust and suitable for larger-scale deployments. Advanced features like user authentication, automated notifications, and analytics on bidding trends could further enrich its capabilities, making it viable for professional use.

REFERENCES:

Books

"Head First Java" by Kathy Sierra and Bert Bates

- This book offers a great introduction to Java, including GUI programming with AWT and Swing, which is helpful for understanding event-driven programming and building applications like the auction system.

YouTube Links

[Java GUI Programming Tutorial \(AWT & Swing\)](#)

- A tutorial that covers how to build simple GUI applications in Java using AWT and Swing. This could be helpful for understanding the event-driven architecture in your code.

APPENDIX

SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;

import java.util.ArrayList;
import javax.swing.Timer;

public class AuctionSystem {
    public static void main(String[] args) {
        new UserSelectionScreen(); // Start with the User Selection screen
    }
}

class UserSelectionScreen {
    private Frame userSelectionFrame;
    private Button auctionerButton, participantButton;

    public UserSelectionScreen() {
        userSelectionFrame = new Frame("Select User Type");
        userSelectionFrame.setLayout(new FlowLayout());

        auctionerButton = new Button("Auctioner");
        participantButton = new Button("Participant");

        userSelectionFrame.add(auctionerButton);
        userSelectionFrame.add(participantButton);

        auctionerButton.addActionListener(e -> new
LoginScreen("Auctioner", userSelectionFrame));
        participantButton.addActionListener(e -> new
LoginScreen("Participant", userSelectionFrame));

        userSelectionFrame.setSize(300, 150);
        userSelectionFrame.setVisible(true);

        userSelectionFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }
}
```

```

    }

    class AuctionerScreen {
        private Frame auctionerFrame;
        private TextField itemNameField, priceField, auctionTimeField;
        private Label timerLabel, statusLabel, bidListLabel, bidCountLabel,
winnerLabel;
        private Button addItemButton, startAuctionButton,
viewParticipantsButton;
        private Timer timer;
        private int timeLeft;

        private static ArrayList<Bid> bids = new ArrayList<>();
        private static String itemName = "No Auction Item";
        private static String basePrice = "0";
        private static boolean auctionRunning = false;

        public AuctionerScreen() {
            auctionerFrame = new Frame("Auctioner Panel");
            auctionerFrame.setLayout(new GridLayout(11, 2, 10, 10));

            itemNameField = new TextField(20);
            priceField = new TextField(20);
            auctionTimeField = new TextField(20);

            timerLabel = new Label("Timer:----- ", Label.RIGHT);
            statusLabel = new Label("");
            bidListLabel = new Label("Bids: ");
            bidCountLabel = new Label("Total Bids: 0");
            winnerLabel = new Label("Winner: --");

            addItemButton = new Button("Add Item");
            startAuctionButton = new Button("Start Auction");
            viewParticipantsButton = new Button("View Participant Details");

            auctionerFrame.add(new Label("Item Name:"));
            auctionerFrame.add(itemNameField);
            auctionerFrame.add(new Label("Base Price:"));
            auctionerFrame.add(priceField);
            auctionerFrame.add(new Label("Auction Time (seconds:)"));
            auctionerFrame.add(auctionTimeField);
            auctionerFrame.add(addItemButton);
            auctionerFrame.add(startAuctionButton);

```

```

    auctionerFrame.add(viewParticipantsButton);
    auctionerFrame.add(timerLabel);
    auctionerFrame.add(new Label());
    auctionerFrame.add(bidListLabel);
    auctionerFrame.add(bidCountLabel);
    auctionerFrame.add(new Label());
    auctionerFrame.add(winnerLabel);
    auctionerFrame.add(statusLabel);

    addItemButton.addActionListener(e -> {
        itemName = itemNameField.getText();
        basePrice = priceField.getText();

        if (!itemName.isEmpty() && !basePrice.isEmpty()) {
            statusLabel.setText("Item added successfully!");
            bids.clear();
            updateBidInfo();
            auctionRunning = true;
            winnerLabel.setText("Winner: --");
        } else {
            statusLabel.setText("Please provide both item name and base
price.");
        }
    });

    startAuctionButton.addActionListener(e -> {
        try {
            timeLeft = Integer.parseInt(auctionTimeField.getText());
            startTimer();
        } catch (NumberFormatException ex) {
            statusLabel.setText("Please enter a valid auction time.");
        }
    });

    viewParticipantsButton.addActionListener(e -> new
ParticipantDetailsScreen(bids));

    auctionerFrame.setSize(500, 500);
    auctionerFrame.setVisible(true);

    auctionerFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            auctionerFrame.dispose();
        }
    });

```

```

    }
    });
}

private void startTimer() {
    if (timer != null) {
        timer.stop();
    }
    timer = new Timer(1000, e -> {
        if (timeLeft > 0) {
            timeLeft--;
            timerLabel.setText("Timer: " + timeLeft + "s");
        } else {
            timer.stop();
            auctionRunning = false;
            timerLabel.setText("Auction Finished!");
            declareWinner();
        }
    });
    timer.start();
}

private void declareWinner() {
    if (bids.isEmpty()) {
        statusLabel.setText("No bids received. Item not sold.");
        winnerLabel.setText("Winner: --");
    } else {
        Bid highestBid = bids.stream().max((b1, b2) ->
Double.compare(b1.getAmount(), b2.getAmount())).get();
        statusLabel.setText("Item sold to " + highestBid.getName() + "
for " + highestBid.getAmount() + ".");
        winnerLabel.setText("Winner: " + highestBid.getName());
        showWinnerPopup(highestBid);
    }
}

private void showWinnerPopup(Bid highestBid) {
    Frame winnerPopup = new Frame("Winner Announcement");
    winnerPopup.setLayout(new FlowLayout());

    Label winnerLabel = new Label("Congratulations, " +
highestBid.getName() + "! You won the auction for " + itemName + " with a
bid of " + highestBid.getAmount() + ".");

```

```

winnerPopup.add(winnerLabel);

winnerPopup.setSize(400, 200);
winnerPopup.setVisible(true);

winnerPopup.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        winnerPopup.dispose();
    }
});
}

public static boolean isAuctionRunning() {
    return auctionRunning;
}

public static String getItemName() {
    return itemName;
}

public static String getBasePrice() {
    return basePrice;
}

public static void addBid(String participantName, double bidAmount)
{
    bids.add(new Bid(participantName, bidAmount));
}

public void updateBidInfo() {
    bidListLabel.setText("Bids: " + (bids.isEmpty() ? "No bids yet" :
bids.get(bids.size() - 1).toString()));
    bidCountLabel.setText("Total Bids: " + bids.size());
}
}

class ParticipantDetailsScreen {
    private Frame participantDetailsFrame;

    public ParticipantDetailsScreen(ArrayList<Bid> bids) {
        participantDetailsFrame = new Frame("Participant Details");
        participantDetailsFrame.setLayout(new GridLayout(bids.size() + 1,
1));
    }
}

```

```

        participantDetailsFrame.add(new Label("Participant Name - Bid
Amount"));

        for (Bid bid : bids) {
            participantDetailsFrame.add(new Label(bid.toString()));
        }

        participantDetailsFrame.setSize(400, 300);
        participantDetailsFrame.setVisible(true);

        participantDetailsFrame.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent we) {
        participantDetailsFrame.dispose();
    }
});
    }
}

```

```

class LoginScreen {
    private Frame loginFrame;
    private TextField usernameField, passwordField;
    private Button loginButton, cancelButton;
    private Label statusLabel;
    private String userType;
    private Frame userSelectionFrame;

    public LoginScreen(String userType, Frame userSelectionFrame) {
        this.userType = userType;
        this.userSelectionFrame = userSelectionFrame;

        loginFrame = new Frame(userType + " Login");
        loginFrame.setLayout(new GridLayout(4, 2, 10, 10));

        usernameField = new TextField(20);
        passwordField = new TextField(20);
        passwordField.setEchoChar('*');
        loginButton = new Button("Login");
        cancelButton = new Button("Cancel");
        statusLabel = new Label("");

        loginFrame.add(new Label("Username:"));
    }
}

```

```

loginFrame.add(usernameField);
loginFrame.add(new Label("Password:"));
loginFrame.add(passwordField);
loginFrame.add(loginButton);
loginFrame.add(cancelButton);
loginFrame.add(statusLabel);

loginButton.addActionListener(e -> {
    String username = usernameField.getText();
    String password = passwordField.getText();

    if (userType.equals("Auctioner")) {
        if (username.equalsIgnoreCase("ADMIN") &&
password.equalsIgnoreCase("ADMIN")) {
            new AuctionerScreen();
        } else {
            statusLabel.setText("Retry: Wrong username/password.");
        }
    } else {
        new ParticipantScreen();
    }
});

cancelButton.addActionListener(e -> {
    userSelectionFrame.setVisible(true);
    loginFrame.setVisible(false);
});

loginFrame.setSize(300, 200);
loginFrame.setVisible(true);

loginFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}
}

class ParticipantScreen {
    private Frame participantFrame;
    private TextField participantNameField, bidAmountField;
    private Button placeBidButton;

```



```

private Label auctionDetailsLabel, statusLabel;

public ParticipantScreen() {
    participantFrame = new Frame("Participant Panel");
    participantFrame.setLayout(new GridLayout(5, 2, 10, 10));

    participantNameField = new TextField(20);
    bidAmountField = new TextField(20);
    placeBidButton = new Button("Place Bid");
    auctionDetailsLabel = new Label("Auction Item: " +
AuctionerScreen.getItemName() + " | Base Price: " +
AuctionerScreen.getBasePrice());
    statusLabel = new Label("");

    participantFrame.add(auctionDetailsLabel);
    participantFrame.add(new Label("Your Name:"));
    participantFrame.add(participantNameField);
    participantFrame.add(new Label("Bid Amount:"));
    participantFrame.add(bidAmountField);
    participantFrame.add(placeBidButton);
    participantFrame.add(statusLabel);

    placeBidButton.addActionListener(e -> {
        String participantName = participantNameField.getText();
        double bidAmount;

        try {
            bidAmount = Double.parseDouble(bidAmountField.getText());

            if (AuctionerScreen.isAuctionRunning()) {
                if (bidAmount >
Double.parseDouble(AuctionerScreen.getBasePrice())) {
                    AuctionerScreen.addBid(participantName, bidAmount);
                    statusLabel.setText("Bid placed successfully!");
                } else {
                    statusLabel.setText("Bid must be greater than the base
price.");
                }
            } else {
                statusLabel.setText("Auction not running.");
            }
        } catch (NumberFormatException ex) {
            statusLabel.setText("Please enter a valid bid amount.");
        }
    });
}

```

```

    }
});

participantFrame.setSize(400, 300);
participantFrame.setVisible(true);

participantFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}
}

class Bid {
    private String name;
    private double amount;

    public Bid(String name, double amount) {
        this.name = name;
        this.amount = amount;
    }

    public String getName() {
        return name;
    }

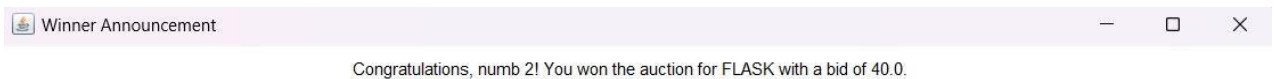
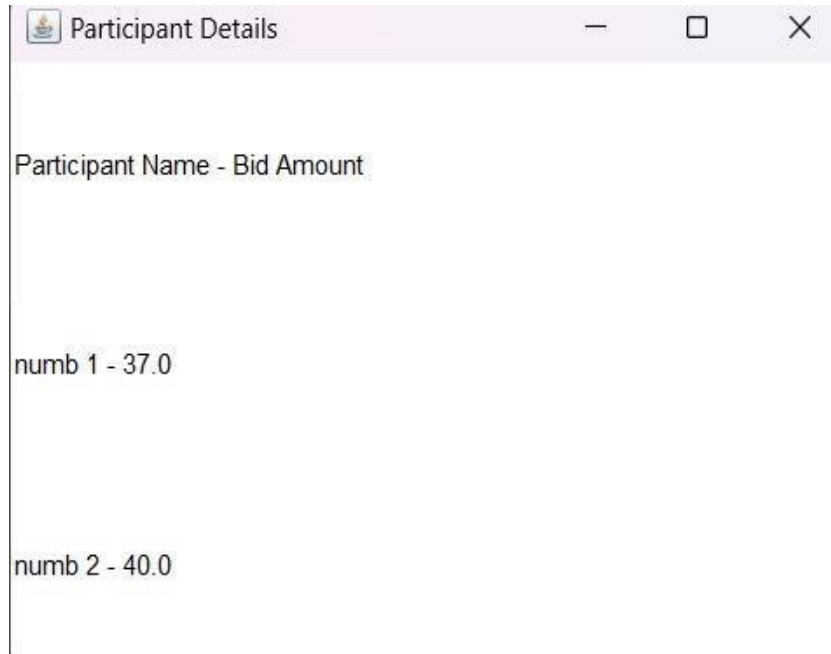
    public double getAmount() {
        return amount;
    }

    @Override
    public String toString() {
        return name + " - " + amount;
    }
}

```

APPENDIX B

SCREENSHOTS



Auctioner Panel

Item Name:

FLASK

Base Price:

30

Auction Time (seconds):

90

Add Item

Start Auction

View Participant Details

Timer: 77s

Bids: No bids yet

Total Bids: 0

Winner: --

Item added successfully!