AUCTION MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

HARIHAR R - (2303811710421051)

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 – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"AUCTION MANAGEMENT SYSTEM "** is the bonafide work of **HARIHAR R - (2303811710421051)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. M. Saravanan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

INTERNAL EXAMINER 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

HARIHAR R

Place: Samayapuram

Date: 02.12.2024

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 am 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 me the opportunity to frame the project with full satisfaction.

I wholeheartedly thank **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encouragement in pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.,** 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 the 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

PEO 1: Domain Knowledge

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

PEO 2: Employability Skills and Research

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

PEO 3: 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 Auction Management System is a comprehensive software application developed using Java's Abstract Window Toolkit (AWT) and Swing libraries, designed to streamline the process of managing and conducting auctions in a user-friendly, efficient manner. This system caters to both auctioneers and bidders, providing an interactive graphical user interface (GUI) for auction-related operations. It includes features for user registration, item listing, bidding, and auction closing, ensuring secure and transparent transactions. The system leverages Swing components like JFrame, JPanel, JButton, JLabel, JTextField, JTable, and JList to create dynamic and intuitive interfaces, enabling users to seamlessly navigate through various auction phases. AWT is utilized for handling event-driven programming, ensuring responsive user interactions. The application implements robust data validation techniques to prevent invalid inputs and employs file handling or database integration for persistent storage of user and auction data. This system not only automates routine tasks such as bid comparison and winner determination but also enhances the overall auction experience by providing real-time updates and notifications. The Auction Management System is highly customizable and can be tailored to fit diverse auction scenarios, ranging from traditional auctions to online platforms, making it a versatile solution for modern auction requirements.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The Auction Management System is a comprehensive software application developed using Java's Abstract Window Toolkit (AWT) and Swing libraries, designed to streamline the process of managing and conducting auctions in a user-friendly, efficient manner. This system caters to both auctioneers and bidders, providing an interactive graphical user interface (GUI) for auction-related operations. It includes features for user registration, item listing, bidding, and auction closing, ensuring secure and transparent transactions. The system leverages Swing components like JFrame, JPanel, JButton, JLabel, JTextField, JTable, and JList to create dynamic and intuitive interfaces, enabling users to seamlessly navigate through various auction phases. AWT is utilized for handling event-driven programming, ensuring responsive user interactions. The application implements robust data validation techniques to prevent invalid inputs and employs file handling or database integration for persistent storage of user and auction data. This system not only automates routine tasks such as bid comparison and winner determination but also enhances the overall auction experience by providing real-time updates and notifications. The Auction Management System is highly customizable and can be tailored to fit diverse auction scenarios, ranging from traditional auctions to online platforms, making it a versatile solution for modern auction requirements.	**PO 1 -3** **PO 2 -3** **PO 3 -3** **PO 4 -3** **PO 5 -3** **PO 6 -3** **PO 7 -3** **PO 8 -3** **PO 9 -3** **PO 10 -3** **PO 11-3** **PO 12 -3**	**PSO 1 -3** **PSO 2 -3** **PSO 3 -3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1Objective

The Java program creates an Auction Management System using a Swing-based GUI, designed to facilitate auctions for both sellers and bidders. Sellers can add items by providing the item name and starting price, which are then displayed in a dropdown menu for bidders. Bidders can select an item, input their name, and place bids higher than the current price. The system continuously tracks the highest bid for each item and records the highest bidder. The auction log, displayed in a text area, maintains a real-time record of item additions, bid updates, and completed sales. Sellers can mark an item as sold once the auction ends, removing it from the active list. The program includes robust input validation to ensure that all fields are filled correctly, bid amounts are numeric, and bids are higher than the current price.

1.2Overview

The Java-based Auction Management System uses Swing to create a user-friendly interface for sellers and bidders. Sellers can list items with names and starting prices, which appear in a dropdown menu for bidders. Bidders place higher bids, and the system updates the highest bid and bidder details in real-time. An auction log tracks all activity, and sellers can mark items as sold when the auction ends. The system includes input validation to ensure proper bid amounts and criteria are met, making it a robust solution for managing both small and large auctions efficiently.

1.3 Java Programming Concepts

The Java code demonstrates key programming concepts, including object-oriented principles with the harihar class managing the auction system. It uses encapsulation to store system states privately and exposes functionality through public methods. The program employs collections like HashMap for managing auction items and bids. GUI components like JComboBox, JTextField, and JTextArea from Swing are used for interactive design, with event handling via ActionListener and lambda expressions. Exception handling ensures robust input validation, and layout managers like BorderLayout and GridLayout organize the interface. Overall, the code integrates OOP, collections, event-driven programming, exception handling, and GUI design for a comprehensive auction management system.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed enhancements for the Auction Management System include adding user authentication to distinguish between sellers and bidders, improving security and personalization. Database integration would allow for persistent auction data and session continuity. Real-time notifications would keep bidders updated on auction changes without manual refresh. Advanced analytics could offer sellers insights into bidding trends, and a more dynamic GUI using JavaFX could improve the user experience. Supporting multiple concurrent auctions and implementing proxy bidding would expand the system's scalability. These features would transform the system into a versatile tool for both casual users and professional auctioneers.

2.2 Block Diagram

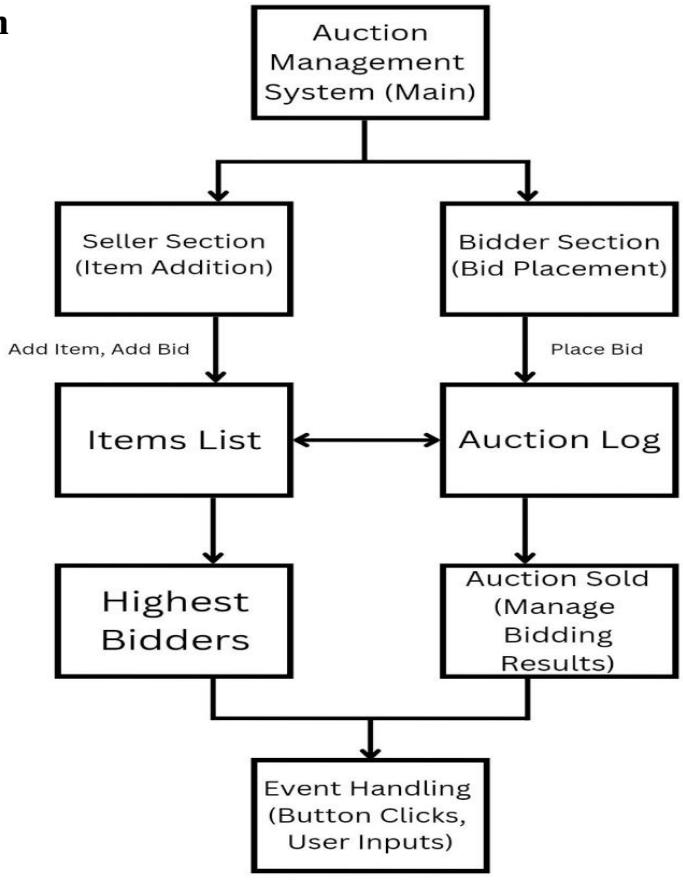


Fig 2.2 Block Diagram

CHAPTER 3
MODULE DESCRIPTION

3.1 Seller Module

The Seller Module is designed to enable sellers to add and manage items for auction. It includes input fields for specifying item names and their starting prices, which are validated to ensure correctness before submission. Sellers can also finalize auctions by selecting an item and marking it as sold, which removes it from the active list of items. The system updates the highest bidder and their winning bid in the auction log upon completion. This module is integrated with the GUI, featuring buttons like "Add Item" and "Sell Item," which trigger respective functionalities. By offering clear input validation and error messaging, the Seller Module ensures smooth operation for users and maintains the integrity of auction data.

3.2 Bidder Module

The Bidder Module allows bidders to view available auction items and place bids. It features a dropdown menu to select items, input fields for bidder names, and bid amounts. This module validates inputs to ensure that bids are numerical and exceed the current highest bid for the selected item. Upon placing a valid bid, the system updates the item's current price, logs the new bid, and tracks the highest bidder. Interactive elements such as "Place Bid" buttons and real-time updates to the GUI enhance the bidding experience. The module provides feedback to bidders about the success or failure of their bids through error messages and auction log entries.

3.3 Auction Log Module

The Auction Log Module acts as the central repository for recording all auction activities, including item additions, bid placements, and final sales. It utilizes a scrollable text area to display real-time updates, ensuring transparency and accountability throughout the auction process. This module is designed to be read-only for users, preventing accidental modifications while maintaining a chronological record of actions. Sellers and bidders rely on the auction log to monitor the progress and outcomes of auctions, making it an essential component for communication and transparency within the system.

3.4 Data Management Module

The Data Management Module underpins the system by using HashMap structures to store and manage auction data. One HashMap tracks the items and their respective prices, while another records the highest bidders for each item. This module ensures efficient retrieval and updating of data, allowing the system to handle dynamic user interactions seamlessly. It also interacts with the Seller and Bidder modules to synchronize updates, such as modifying prices or removing items from the list once sold. By encapsulating the core data handling logic, this module ensures the system remains consistent and scalable.

3.5 GUI Module

The GUI Module provides the interactive interface through which users interact with the system. Built using the Swing framework, it includes components like JTextField, JTextArea, JComboBox, and buttons to facilitate user input and display auction data. Layout managers like BorderLayout and GridLayout organize these components into Seller, Bidder, and Log sections for intuitive navigation. The module also incorporates dialog boxes (JOptionPane) to deliver feedback and handle errors gracefully. This user-centric design ensures that the system is accessible, engaging, and visually structured to enhance the overall auction management experience.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Auction Management System built with Java provides a comprehensive platform for managing online auctions, showcasing key programming concepts like object-oriented design, event-driven programming, and data structures like HashMap. The modular design allows users to easily add items, place bids, and finalize sales while maintaining an organized and responsive system. The Seller and Bidder sections offer intuitive functionality, with real-time updates on bids and prices, and the Auction Log ensures transparency by recording all auction activities. Exception handling ensures system reliability by managing invalid user inputs. While the system covers basic auction operations, there is potential for future enhancements, such as real-time notifications, multi-auction support, and an improved user interface. These improvements would make the system more versatile and scalable for larger operations. Overall, this project serves as a strong foundation for learning Java programming and provides a dynamic and interactive auction system with clear paths for further development.

4.2 FUTURE SCOPE

The future scope of the Auction Management System includes several enhancements to improve scalability and user experience. Integrating user authentication and roles would allow personalized experiences for sellers, bidders, and administrators. Real-time bidding updates and notifications could increase engagement. Database integration (e.g., MySQL or MongoDB) would enable persistent data and session continuity, scaling the system to handle large numbers of users and items. Multi-auction support would allow concurrent auctions, catering to various events. Proxy bidding and advanced search/filter features would enhance the bidding experience. A modern, responsive GUI using JavaFX or web technologies could improve user interaction. Analytics and reporting tools would provide sellers with insights into bid trends and auction performance. Payment gateway integration could streamline transactions after auctions. These enhancements would transform the system into a robust, scalable platform capable of handling large-scale auctions, benefiting both casual users and professional auctioneers.

APPENDIX A
(SOURCE CODE)

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.util.HashMap;

public class Harihar {

    private JFrame frame;

    private JTextField itemNameField, itemPriceField, bidAmountField, bidderNameField;

    private JTextArea auctionLog;

    private JComboBox<String> itemComboBox;

    private HashMap<String, Integer> items;

    private HashMap<String, String> highestBidders;

    public Harihar() {

        items = new HashMap<>();

        highestBidders = new HashMap<>();

        initialize();

    }


    private void initialize() {

        frame = new JFrame("Auction Management System");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
frame.setSize(600, 600);

frame.setLayout(new BorderLayout());


// Title Label

JLabel titleLabel = new JLabel("Auction Management System", JLabel.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

frame.add(titleLabel, BorderLayout.NORTH);


// Seller Panel

JPanel sellerPanel = new JPanel(new GridLayout(5, 2, 5, 5));

sellerPanel.setBorder(BorderFactory.createTitledBorder("Seller Section"));

sellerPanel.add(new JLabel("Item Name:"));

itemNameField = new JTextField();

sellerPanel.add(itemNameField);

sellerPanel.add(new JLabel("Starting Price:"));

itemPriceField = new JTextField();

sellerPanel.add(itemPriceField);

JButton addItemButton = new JButton("Add Item");

addItemButton.addActionListener(this::addItem);

sellerPanel.add(addItemButton);

JButton sellItemButton = new JButton("Sell Item");
```

```
sellItemButton.addActionListener(this::sellItem);
sellerPanel.add(sellItemButton);
frame.add(sellerPanel, BorderLayout.WEST);

// Bidder Panel
JPanel bidderPanel = new JPanel(new GridLayout(4, 2, 5, 5));
bidderPanel.setBorder(BorderFactory.createTitledBorder("Bidder Section"));
bidderPanel.add(new JLabel("Select Item:"));
itemComboBox = new JComboBox<>();
bidderPanel.add(itemComboBox);
bidderPanel.add(new JLabel("Bidder Name:"));
bidderNameField = new JTextField();
bidderPanel.add(bidderNameField);
bidderPanel.add(new JLabel("Bid Amount:"));
bidAmountField = new JTextField();
bidderPanel.add(bidAmountField);
JButton placeBidButton = new JButton("Place Bid");
placeBidButton.addActionListener(this::placeBid);
bidderPanel.add(placeBidButton);
frame.add(bidderPanel, BorderLayout.EAST);
```

```
        // Auction Log
        auctionLog = new JTextArea();
        auctionLog.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(auctionLog);
        scrollPane.setBorder(BorderFactory.createTitledBorder("Auction Log"));
        frame.add(scrollPane, BorderLayout.CENTER);
        frame.setVisible(true);
    }
    private void addItem(ActionEvent e) {
        String itemName = itemNameField.getText().trim();
        String priceText = itemPriceField.getText().trim();
        if (itemName.isEmpty() || priceText.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Item name and price are required!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try {
            int startingPrice = Integer.parseInt(priceText);
            items.put(itemName, startingPrice);
            itemComboBox.addItem(itemName);
            auctionLog.append("Item added: " + itemName + " (Starting Price: $" + startingPrice + ")\n");
```

```
            itemNameField.setText("");
            itemPriceField.setText("");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Price must be a number!", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void placeBid(ActionEvent e) {
        String selectedItem = (String) itemComboBox.getSelectedItem();
        String bidderName = bidderNameField.getText().trim();
        String bidText = bidAmountField.getText().trim();
        if (selectedItem == null || bidderName.isEmpty() || bidText.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "All fields are required!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try {
            int bidAmount = Integer.parseInt(bidText);
            int currentPrice = items.get(selectedItem);
            if (bidAmount > currentPrice) {
                items.put(selectedItem, bidAmount);
```

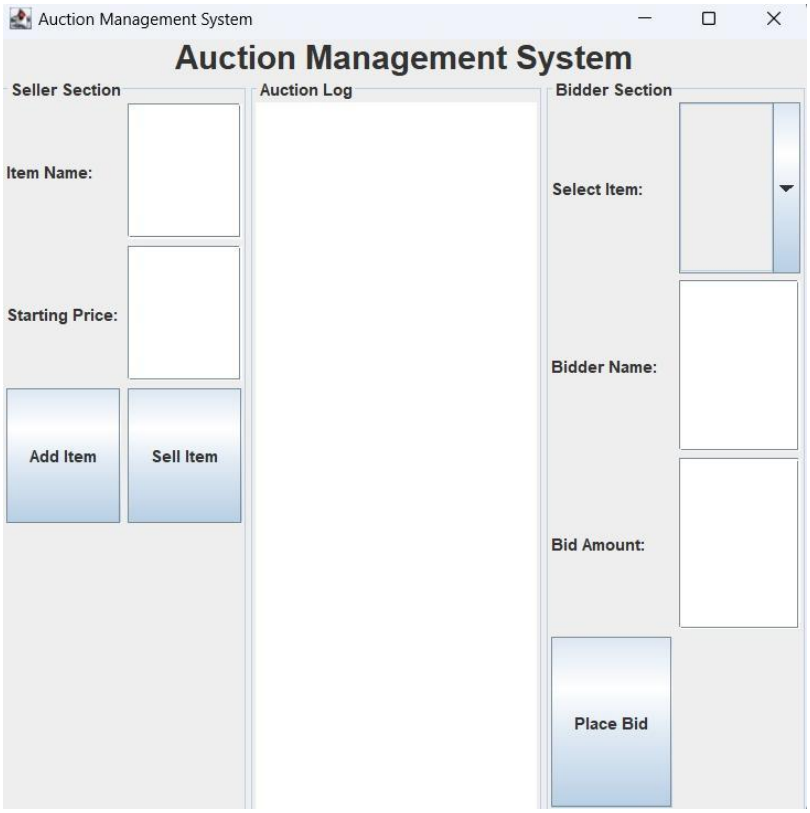
```
                highestBidders.put(selectedItem, bidderName);
                auctionLog.append("New bid for " + selectedItem + ": $" + bidAmount + " by " + bidderName + "\n");
                bidAmountField.setText("");
                bidderNameField.setText("");
            } else {
                JOptionPane.showMessageDialog(frame, "Bid must be higher than the current price!", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Bid amount must be a number!", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    private void sellItem(ActionEvent e) {
        String selectedItem = (String) itemComboBox.getSelectedItem();
        if (selectedItem == null || !items.containsKey(selectedItem)) {
            JOptionPane.showMessageDialog(frame, "No item selected or item not found!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String highestBidder = highestBidders.get(selectedItem);
        int finalPrice = items.get(selectedItem);
```

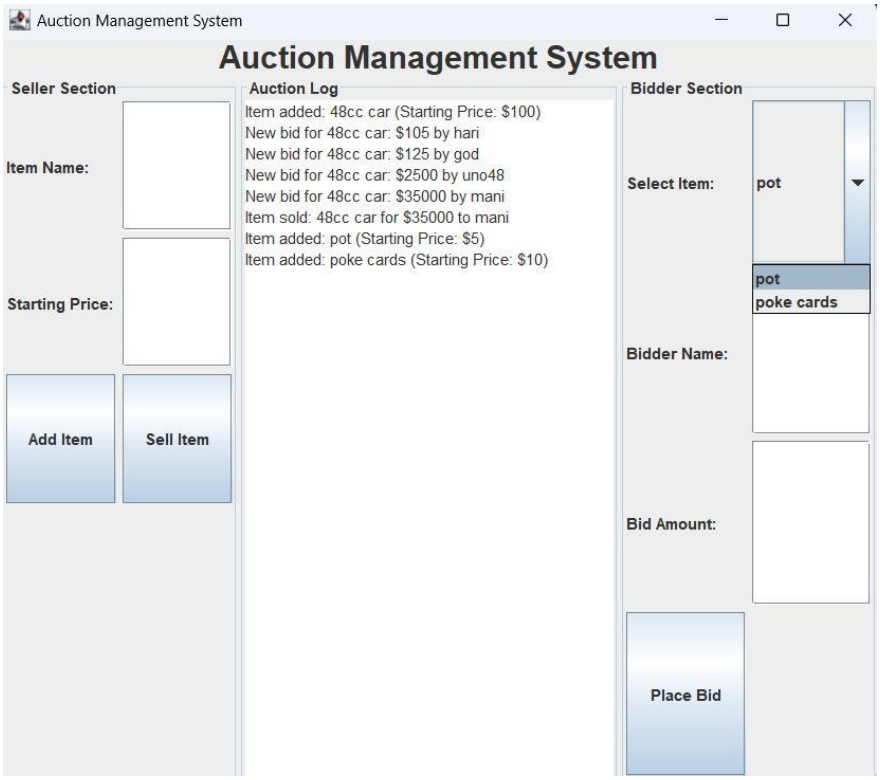
```
        if (highestBidder == null) {
            JOptionPane.showMessageDialog(frame, "No bids placed for this item!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        auctionLog.append("Item sold: " + selectedItem + " for $" + finalPrice + " to " + highestBidder + "\n");
        items.remove(selectedItem);
        highestBidders.remove(selectedItem);
        itemComboBox.removeItem(selectedItem);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Harihar::new);
    }
}
```

APPENDIX B

(OUTPUT)





REFERENCES

1. Java Official Documentation (Oracle)

Oracle's official documentation provides comprehensive information on Java programming, including Swing components, event handling, and HashMaps.

URL: https://docs.oracle.com/en/java/

2. Swing GUI Framework – Java Tutorials (Oracle)

The Oracle tutorials for Swing offer detailed guidance on building graphical user interfaces in Java, including components like JFrame, JTextField, JTextArea, JComboBox, and layout managers.

URL: https://docs.oracle.com/javase/tutorial/uiswing/

3. HashMap Class – Java Documentation (Oracle)

HashMap is used in the Auction Management System to store auction items and track highest bidders. This official Java documentation explains the use of HashMap.

URL: https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html