K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A Project Report

on

AUCTION MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Ms. Hema R., M.E.,

Assistant Professor / Information Technology

Submitted By

KANISHKA L (ECA23049)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(Autonomous)

TRICHY - 621112

DECEMBER 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY - 621112

BONAFIDE CERTIFICATE

Certified that this project report on **"AUCTION MANAGEMENT SYSTEM"** is the Bonafide work of **KANISHKA L (2303811710622049)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Ms. HEMA R., M.E.,

SUPERVISOR,

Department of Information Technology,

K. Ramakrishnan College of Technology,

Trichy - 621112

Signature

Dr. SYEDAKBAR S., M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of ECE,

K. Ramakrishnan College of Technology,

Trichy - 621112

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To create innovative and socially responsible Electronics and Communication Engineers with design skills and research focus to meet Societal and Industrial needs.

MISSION OF THE DEPARTMENT

- M1: To provide high quality education and professional ethics to students through enhanced learning environment
- M2: To impart a creative environment towards centre of excellence in department with design skill and exposure for research.
- M3: To nurture required employable skills of students to satisfy the industry and social needs with ethical and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- PEO1: Core Knowledge Development: Graduates will have enhanced engineering skills in the field of electronics, communication and interdisciplinary areas to serve the society with global standards.

- PEO2: Professional development: Graduates will apply the technical knowledge for continuous up gradation of their professional skills to become an inimitable employee, researcher or entrepreneur.
- PEO3: Analytical Thinking: Graduates will have analytic and thinking skills to provide the innovative solutions for industry and societal requirements.

PROGRAM OUTCOMES

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 modelling 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- PSO1: To analyse, design and develop solutions by applying foundational concepts of electronics and communication engineering.
- PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

ABSTRACT

The Auction Management System (AMS) is an online platform designed to facilitate seamless auctions for various categories of items, including collectibles, antiques, real estate, and more. Utilizing AWT (Abstract Window Toolkit) and Swing for the user interface, this system enables sellers to list their items, manage the auction process, and engage with potential buyers. Bidders can place bids on items, track auction progress, and receive real-time updates. The system ensures transparency by providing clear bidding history, item details, and auction timelines, while maintaining security through user authentication and bid validation. Sellers can set reserve prices, view bidder activity, and determine the winner once the auction ends. The AMS aims to create a fair, efficient, and user-friendly auction environment by combining AWT and Swing for a responsive and visually appealing interface. With real-time bid tracking and robust functionality, it offers a competitive edge for sellers and buyers in an online auction setting, ensuring a secure, reliable, and transparent experience for all participants.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
Demonstrate knowledge of Java swing for developing interactive GUIs.	**PO 1**	**3**
Design and implementation modular systems for auction management	**PO 2**	**1**
Evaluate and ensure the functionality of the application through testing and designing	**PO 5**	**2**

Note: 1- Low, 2 – Medium, 3 - High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Auction Management System using Swing for the graphical user interface. It allows users to add auction items, place bids, and view auction details. Items can be added with a specified starting price, and users can bid with validation to ensure bids exceed both the starting price and any current highest bid. The system displays all auction items along with their highest bids, if any, providing a clear overview of the auction status. The user-friendly interface ensures easy interaction for managing auctions effectively.

1.1 Objective

The primary objective of the Auction Management System is to create an intuitive and secure platform for conducting online auctions. It aims to provide sellers with the ability to list items for auction, allow bidders to place competitive bids, and manage the auction lifecycle efficiently. The system ensures transparency, fair competition, and satisfaction for both buyers and sellers by integrating real-time updates, validation, and a user-friendly interface using Java's AWT and Swing.

1.2 Overview

The Auction Management System will consist of three core.Sellers can list items for auction, providing details such as item name, description, starting price, and seller's name.Bidders can browse listed items and place bids.The system enforces rules such as bids must be higher than the starting price or the current highest bid.Displays all listed items, their starting prices, and current highest bids.Ensures visibility of auction

1.3 Java Programming Concepts

❖ Object-Oriented Programming (OOP):

The program uses classes (Item, Bid, and AuctionSystem) to encapsulate data and behaviors.Concepts like encapsulation, inheritance (implied by using Object methods like toString()), and modular design are showcased.

❖ Graphical User Interface (GUI) Programming:

Swing is used for GUI components (JFrame, JPanel, JButton, and JOptionPane).Event handling is implemented using ActionListener to handle button clicks.

❖ Error Handling:

Input validation ensures correct data types (e.g., parsing Double values).Error handling is implemented using try-catch blocks to manage invalid inputs.

❖ Data Structures:

The program uses a List (from the ArrayList class) to store

and manage multiple Item objects.The findItemByName method illustrates searching through a list.

❖ User Interaction and Dialogs:

Interactive dialogs (JOptionPane) are used for adding items, placing bids, and displaying information.Dynamic messages provide real-time feedback based on user inputs and auction state.

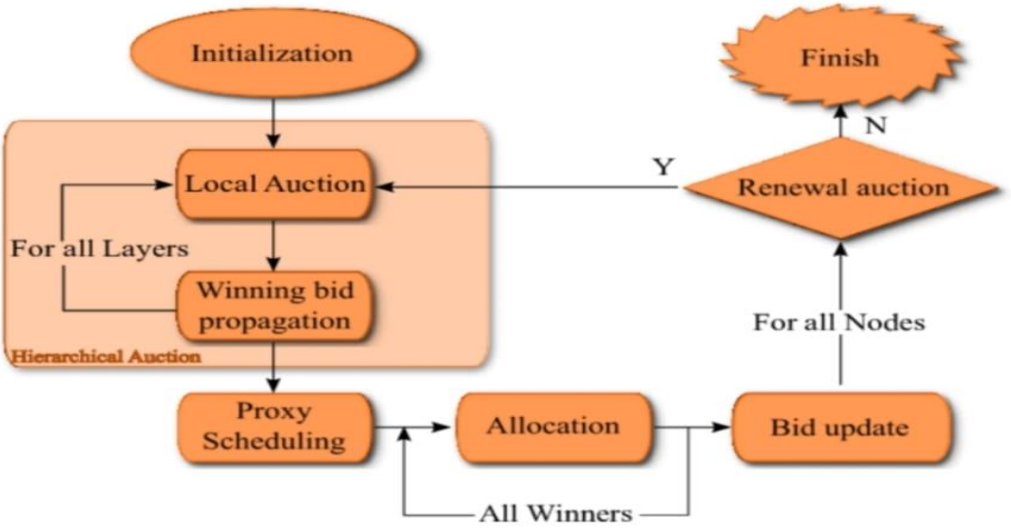
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work involves creating an Auction Management System using Java's AWT and Swing for the user interface. Sellers can list items for auction by entering details like item name, description, and starting price. Bidders can view the listed items, place bids higher than the current highest bid, and track the auction's progress. The system will handle hierarchical auctions, propagating winning bids and ensuring proxy scheduling for fair allocation. Renewal auctions will allow relisting unsold items, and the system will dynamically update bids and allocate items to winners

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 USER MODULE

The user module of the Auction Management System, implemented using AWT and Swing, will have distinct interfaces for both sellers and bidders. For sellers, the interface will allow them to list items for auction by filling in fields such as item name, description, starting price, and seller's name using JTextField and JTextArea. A "Submit" button (JButton) will trigger the listing process and display the items in a table (JTable). Bidders can browse the listed items, see details such as the current bid, and place their own bids, with the system ensuring that bids exceed the current highest bid. The interface will include dynamic updates to show the current highest bid using JLabel or JTable, and each user will be able to view the auction status. An ActionListener will be used to validate bids and update auction details in real-time. Additionally, the system will provide visibility for all users to view ongoing and completed auctions using JPanel and JTable components

3.2 Iteam Module

The Item module of the Auction Management System, designed using AWT and Swing, will allow both sellers and bidders to interact with auctioned items effectively. For sellers, a user-friendly form (JPanel with JTextField and JTextArea) will facilitate the input of essential item details, including item name, description, starting price, and the seller’s name. A "Submit" button (JButton) will trigger the addition of the item to the auction. The system will use a JTable to display the listed items, showing the item’s name, description, starting price, and the current highest bid.

3.3 Bid Module

The Bid module in the Auction Management System, built with AWT and Swing, allows bidders to view active items in a JTable and place bids via a JTextField. The system ensures that bids exceed the starting price or current highest bid by validating input with an ActionListener. Once a valid bid is placed, the highest bid is updated in real-time, and the status is reflected in the UI. A "Place Bid" button submits bids, with confirmation or alerts shown using JOptionPane.

3.4 Auction Module

The Auction module in the Auction Management System, developed using AWT and Swing, allows sellers to list items with details such as name, description, and starting price through input fields. Bidders can browse the items in a JTable, view current bid statuses, and place higher bids via a JTextField. The system ensures real-time updates of the highest bid and auction status, maintaining visibility for all users. A dynamic interface, including buttons and labels, allows seamless interaction and bid tracking.

3.5 Status Monitoring Module

The Status Monitoring module in the Auction Management System, designed using AWT and Swing, provides real-time tracking of auction statuses. It displays a dynamic list of all active auctions, showing item details, current highest bids, and the auction’s progress. Using components like JTable and JLabel, the system updates the bid status as users place new bids. This module ensures that both sellers and bidders can monitor the auction's current state, ensuring full visibility of ongoing and completed auctions.

CHAPTER 4

RESULTS AND DISCUSSION

This Java program creates a GUI-based auction management system. Users can add items with a starting price, place bids on available items, and view a list of all items along with the current highest bids. It uses classes like Item and Bid to model auction entities and ensures bid validation. The application features a simple interface with buttons for adding items, placing bids, and showing auction details, making it intuitive to manage auctions effectively.

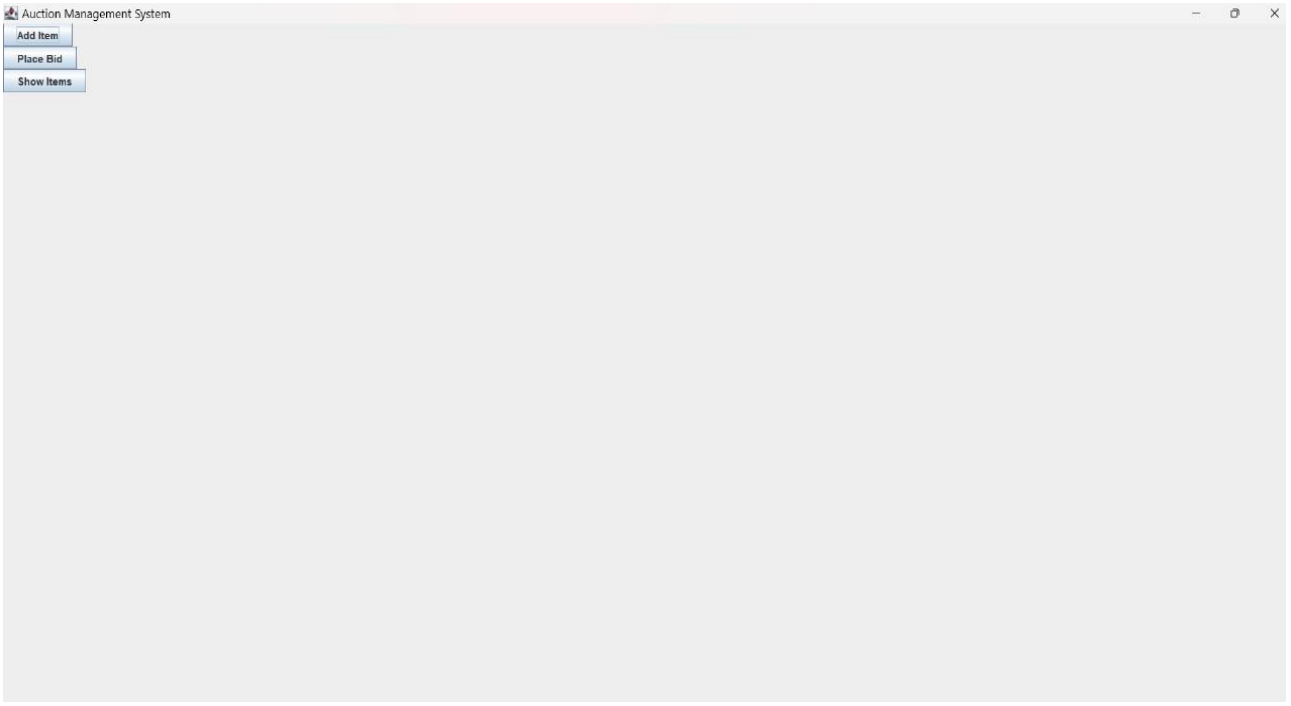


Fig 4.1 Auction Management System

This image shows a basic graphical user interface (GUI) for an auction management system. It features three buttons: "Add Item," "Place Bid," and "Show Items," which likely trigger actions to add new items to the auction, place bids on existing items, and view a list of items and their current bids, respectively.

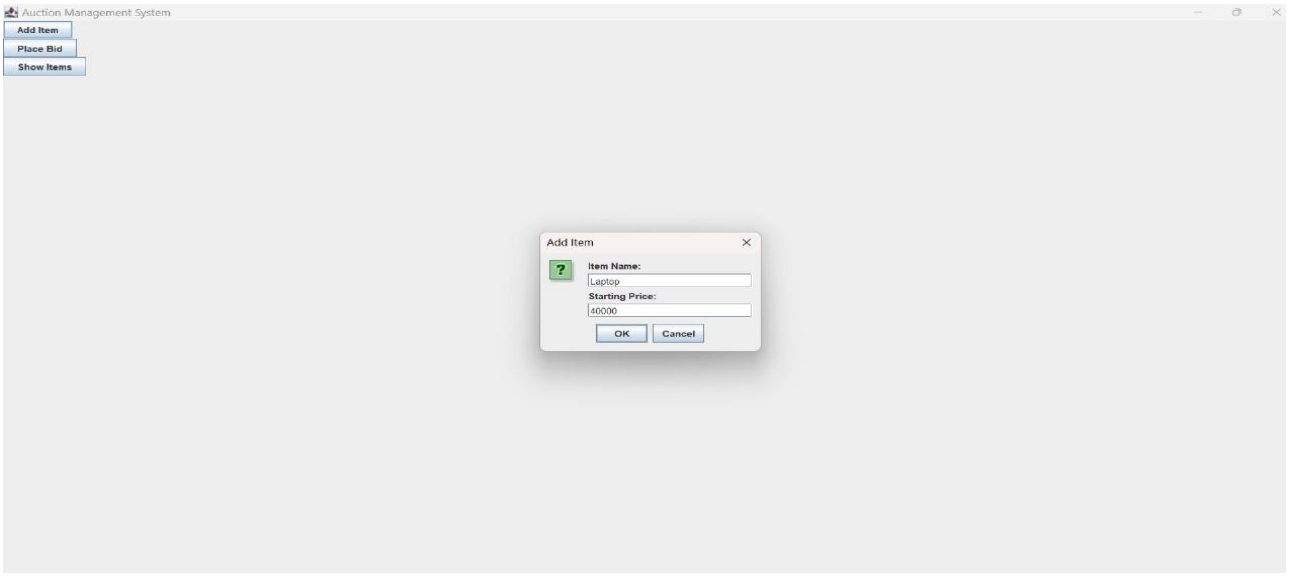


Fig 4.2 Adding an Item to the Auction

This window appears when the "Add Item" button is clicked. It allows the user to enter the details of a new item for auction, including the item name and starting price**.**

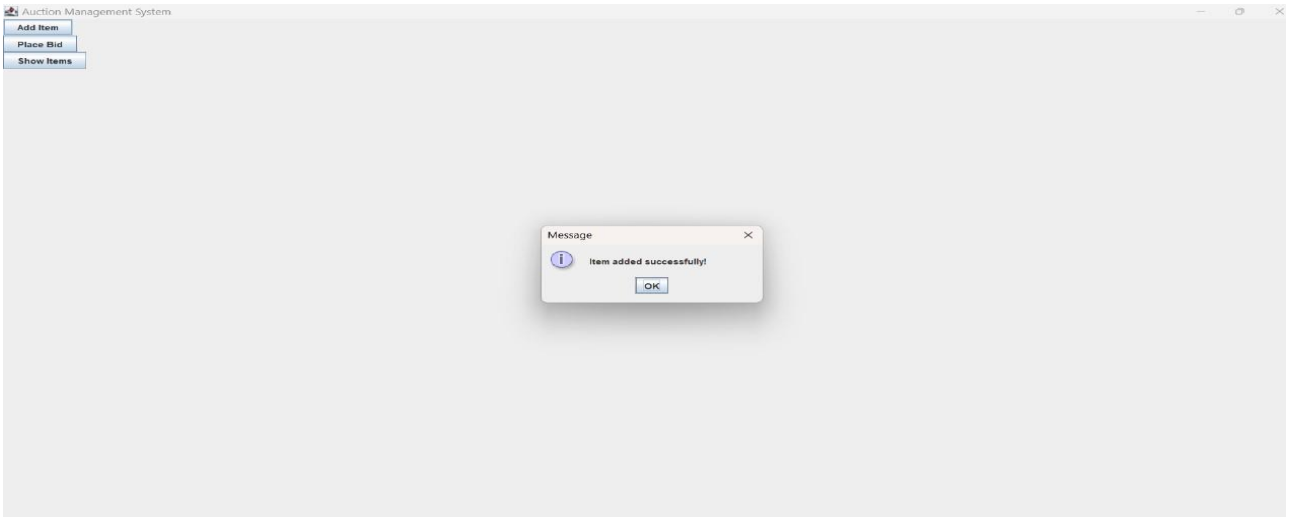


Fig 4.3 Confirmation Message

It informs the user that the item was added successfully and provides an "OK”button to dismiss the message.

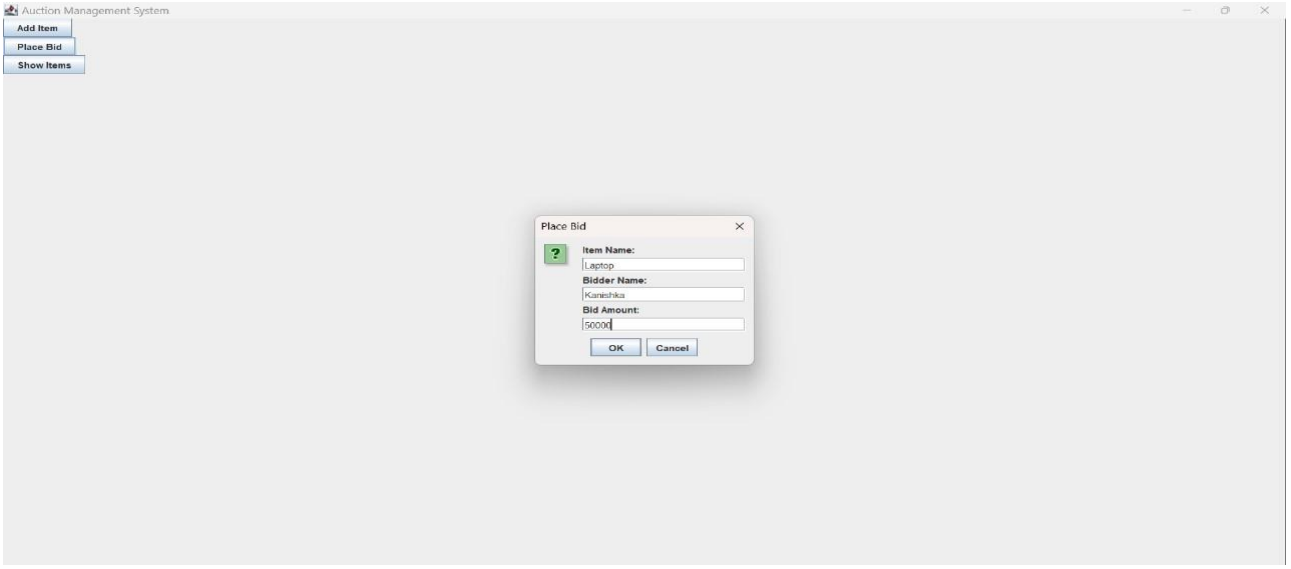


Fig 4.4 Placing a Bid

This image shows a pop-up window within the auction management system. This window appears when the "Place Bid" button is clicked. It allows the user to enter the details of their bid, including the item name, bidder name, and bid amount.

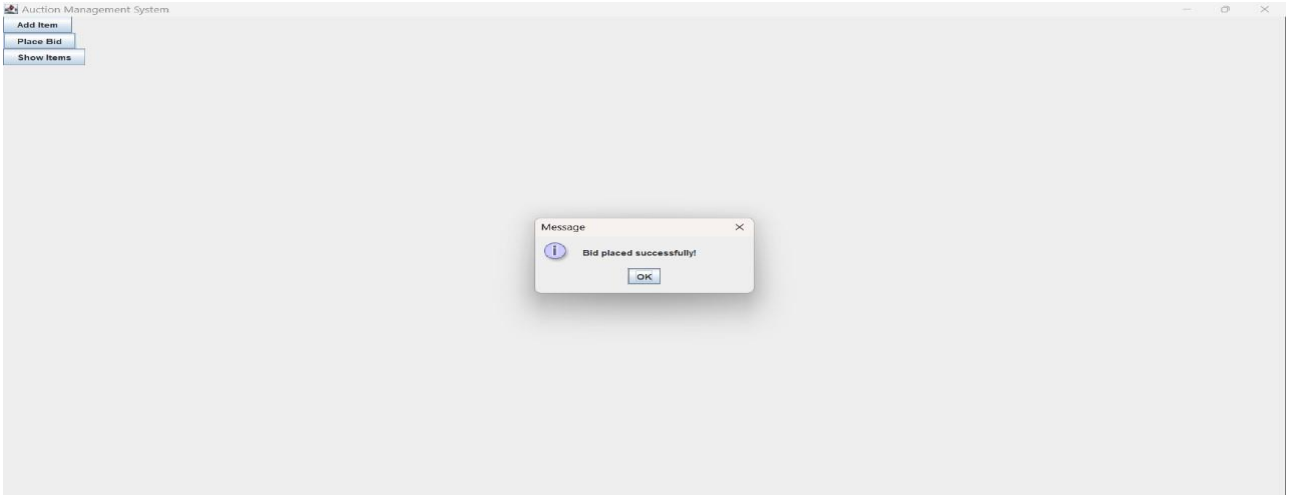


Fig 4.5 Bid Confirmation Message

It informs the user that the bid was placed successfully and provides an "OK" button to dismiss the message.

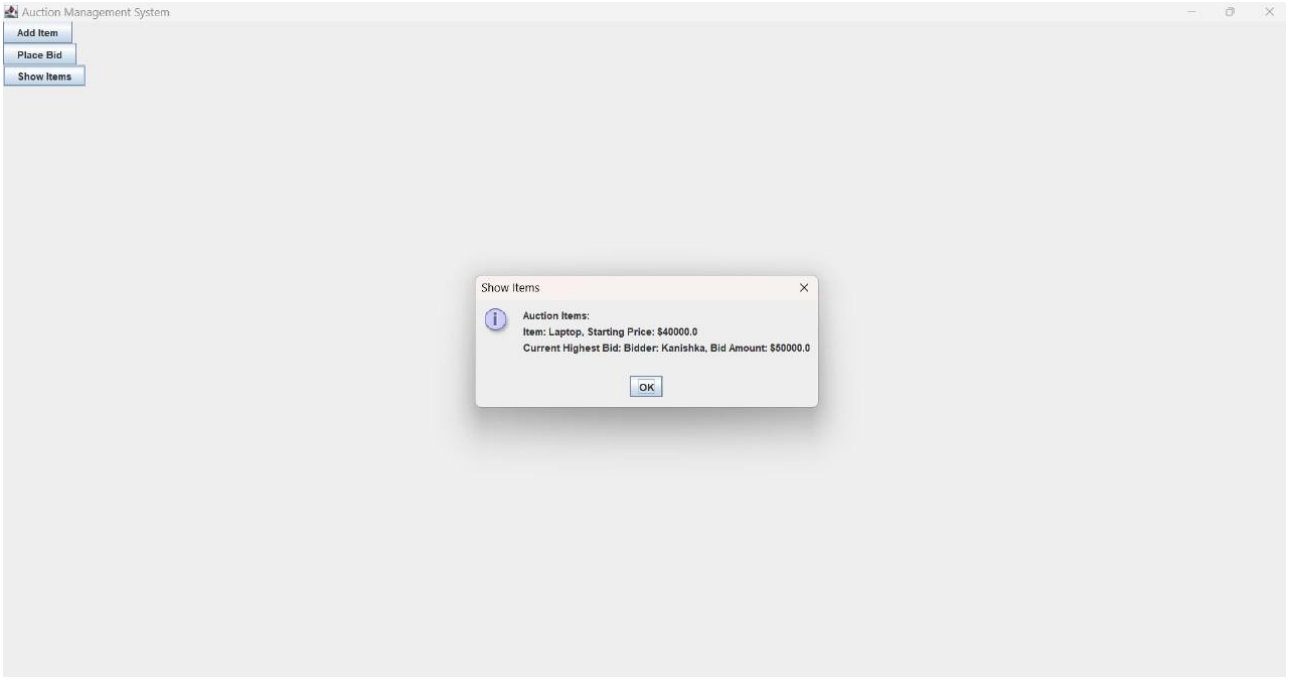


Fig 4.6 Viewing Auction Items

This image shows a pop-up window within the auction management system. This window appears when the "Show Items" button is clicked. It displays a list of items currently up for auction, including their names, starting prices, and the current highest bid for each item.

CHAPTER 5

CONCLUSION

In conclusion, the Auction Management System developed using AWT and Swing offers a robust and intuitive platform for both sellers and bidders, ensuring an efficient and interactive auction experience. Sellers can easily list their items, and bidders can browse, place bids, and monitor the status of each auction in real time. The system effectively enforces bid rules, ensuring fair play and preventing invalid bid submissions. AWT and Swing's lightweight and responsive graphical user interface components facilitate smooth user interactions and immediate feedback on bid placements.

The integration of features like real-time bid updates, visibility of current highest bids, and dynamic auction status displays ensures transparency throughout the auction process. The modular design, divided into core functionalities such as item listing, bidding, and status monitoring, enhances the system’s flexibility and scalability. With clear user interfaces and seamless transitions between auction phases, the system maintains high usability standards. This approach ultimately provides an easy-to-use, reliable, and transparent auction platform, meeting the needs of both sellers and bidders in an efficient manner.

REFERENCES

1. "Design and Implementation of an Auction Management System Using Java" by S. S. Iyengar et al. (2011) - International Journal of Computer Science and Information Security, Vol. 9, No. 5

2. "Auction Management System: A Java-Based Approach" by R. K. Singh et al. (2013) - International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 3

3. "Development of an Auction Management System Using Java Swing" by A. K. Singh et al. (2015) - International Journal of Computer Applications, Vol. 111, No. 11

APPENDIX

(Coding)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;

// Item class to represent an auction item
class Item {
    private String itemName;
    private double startingPrice;
    private Bid highestBid;

    public Item(String itemName, double startingPrice) {
        this.itemName = itemName;
        this.startingPrice = startingPrice;
    }

    public String getItemName() {
        return itemName;
    }

    public double getStartingPrice() {
         return startingPrice;
    }
```

```
 public Bid getHighestBid() {
        return highestBid;
    }

    public void setHighestBid(Bid bid) {
        this.highestBid = bid;
    }

    @Override
    public String toString() {
        return "Item: " + itemName + ", Starting Price: $" + startingPrice;
    }
}

// Bid class to represent a bid
class Bid {
    private String bidderName;
    private double bidAmount;

    public Bid(String bidderName, double bidAmount) {
        this.bidderName = bidderName;
        this.bidAmount = bidAmount;
     }
     public String getBidderName() {
         return bidderName;
     }
```

```
    public double getBidAmount() {
        return bidAmount;
    }

    @Override
    public String toString() {
        return "Bidder: " + bidderName + ", Bid Amount: $" + bidAmount;
    }
}

// AuctionSystem class to manage items and bids
class AuctionSystem {
    private List<Item> items = new ArrayList<>();

    public void addItem(String itemName, double startingPrice) {
        items.add(new Item(itemName, startingPrice));
    }

    public String placeBid(String itemName, String bidderName, double bidAmount)
{
        Item item = findItemByName(itemName);
        if (item == null) {
            return "Bid return "Item not found.";
        }
        if (bidAmount < item.getStartingPrice()) {
must be higher than the starting price.";
```

```
        }
        if (item.getHighestBid() != null && bidAmount <= item.getHighestBid().getBidAmount()) {
            return "Bid must be higher than the current highest bid.";
        }
        item.setHighestBid(new Bid(bidderName, bidAmount));
        return "Bid placed successfully!";
    }

    public List<Item> getItems() {
        return items;
    }

    private Item findItemByName(String itemName) {
        for (Item item : items) {
            if (item.getItemName().equalsIgnoreCase(itemName)) {
                return item;
            }
        }
        return null;
    }
}

// Main GUI class
public class AuctionSystemGUI {
    private AuctionSystem auctionSystem = new AuctionSystem();
```

```
public static void main(String[] args) {
    new AuctionSystemGUI().createAndShowGUI();
}

public void createAndShowGUI() {
    JFrame frame = new JFrame("Auction Management System");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(600, 400);

    JPanel panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    frame.add(panel);

    JButton addItemButton = new JButton("Add Item");
    JButton placeBidButton = new JButton("Place Bid");
    JButton showItemsButton = new JButton("Show Items");

    panel.add(addItemButton);
    panel.add(placeBidButton);
    panel.add(showItemsButton);

    addItemButton.addActionListener(e -> addItemDialog(frame));
    placeBidButton.addActionListener(e -> placeBidDialog(frame));
    showItemsButton.addActionListener(e -> showItemsDialog(frame));

    frame.setVisible(true);
}
```

```
    private void addItemDialog(JFrame frame) {
        JTextField itemNameField = new JTextField();
        JTextField startingPriceField = new JTextField();

        Object[] message = {
            "Item Name:", itemNameField,
            "Starting Price:", startingPriceField,
        };

        int option = JOptionPane.showConfirmDialog(frame, message, "Add Item",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            String itemName = itemNameField.getText();
            double startingPrice;
            try {
                startingPrice = Double.parseDouble(startingPriceField.getText());
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(frame, "Invalid starting price. Please
enter a valid number.");
                return;
            }
            auctionSystem.addItem(itemName, startingPrice);
             JOptionPane.showMessageDialog(frame, "Item added successfully!");
        }
    }
```

```
    private void placeBidDialog(JFrame frame) {
        JTextField itemNameField = new JTextField();
        JTextField bidderNameField = new JTextField();
        JTextField bidAmountField = new JTextField();

        Object[] message = {
            "Item Name:", itemNameField,
            "Bidder Name:", bidderNameField,
            "Bid Amount:", bidAmountField,
        };

        int option = JOptionPane.showConfirmDialog(frame, message, "Place Bid",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            String itemName = itemNameField.getText();
            String bidderName = bidderNameField.getText();
            double bidAmount;
            try {
                bidAmount = Double.parseDouble(bidAmountField.getText());
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(frame, "Invalid bid amount. Please
enter a valid number.");
                return;
            }
            String result = auctionSystem.placeBid(itemName, bidderName,
bidAmount);
            JOptionPane.showMessageDialog(frame, result);
```

```
        }
    }

    private void showItemsDialog(JFrame frame) {
        List<Item> items = auctionSystem.getItems();
        StringBuilder message = new StringBuilder("Auction Items:\n");
        for (Item item : items) {
            message.append(item.toString()).append("\n");
            if (item.getHighestBid() != null) {
                message.append("Current Highest Bid: ").append(item.getHighestBid()).append("\n");
            } else {
                message.append("No bids yet.\n");
            }
            message.append("\n");
        }
        if (items.isEmpty()) {
            message.append("No items available.");
        }
        JOptionPane.showMessageDialog(frame, message.toString(), "Show Items", JOptionPane.INFORMATION_MESSAGE);
    }
}
```