A Project Report

on

AUCTION MANAGEMENT SYSTEM

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

of

**CGB1201 – JAVA PROGRAMMING**

Under the guidance of

### Mrs. I.Karthika M.E.,

### Assistant Professor / CSE

Submitted By



**SASMITHA S (927623BCS103)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

DECEMBER 2024

# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that thisproject report on **“AUCTION MANAGEMENT SYSTEM”** is the bonafide work of **SASMITHA S (927623BCS103)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mr.K.KARTHIK M.E.,** | **Dr. D. PRADEEP M.E.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science and Engineering, | Department of Computer Science and Engineering, |
| M.Kumarasamy College of Engineering, | M.Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |



ii

**DEPARTMENT OF COMPUTER SCIENCE AND 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 achieve education and research excellence in Computer Science and Engineering 

**MISSION OF THE DEPARTMENT**

* To excel in academic through effective teaching learning techniques
* To promote research in the area of computer science and engineering with the focus on innovation
* To transform students into technically competent professionals with societal and ethical responsibilities

**PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

**PEO 1:**Graduates will have successful career in software industries and R&D divisions through continuous learning.

**PEO 2:** Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

**PEO 3:** Graduates will excel in their profession by being ethically and socially responsible.

**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 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.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

* **PSO1: Professional Skills:**  Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.
* **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

# 

# v

# ABSTRACT

The Auction Management System is a comprehensive software solution designed to facilitate online auctions for various items, including collectibles, antiques, and real estate etc. The system enables sellers to list items, allows bidders to place bids, and manages the auction process from start to finish. It ensures transparency, efficiency, and security, providing a fair competition platform for both sellers and buyers. The system incorporates user registration, auction creation, bidding, listing of active auctions, and auction result management.

vi

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The Auction Management System is a comprehensive software solution designed to facilitate online auctions for various items, including collectibles, antiques, and real estate etc. The system enables sellers to list items, allows bidders to place bids, and manages the auction process from start to finish. It ensures transparency, efficiency, and security, providing a fair competition platform for both sellers and buyers. The system incorporates user registration, auction creation, bidding, listing of active auctions, and auction result management. | **PO1,**  **PO2,**  **PO3,**  **PO8,**  **PO9,**  **PO10,**  **PO11** | **PSO1,**  **PSO2,** |

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

**SUPERVISOR HEAD OF THE DEPARTMENT**

vii

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** No. | TITLE | **PAGE** No. |
|  | ABSTRACT | vi |
| 1 | INTRODUCTION | 1 |
|  | 1.1 Objective | 1 |
|  | 1.2 Overview | 1 |
|  | 1.3 Java Programming concepts | 2 |
| 2 | PROJECT METHODOLOGY | 4 |
|  | 2.1Proposed Work | 4 |
|  | 2.2 Block Diagram | 5 |
| 3 | MODULE DESCRIPTION | 6 |
|  | 3.1 User Registration Module | 6 |
|  | 3.2 User Login Module | 6 |
|  | 3.3 Auction Creation Module | 6 |
|  | 3.4 Auction Listing Module | 6 |
|  | 3.5 Bid Placement Module3.6 Result Module | 67 |
| 4 | RESULTS AND DISCUSSION | 8 |
| 5 | CONCLUSION | 13 |
|  | REFERENCES | 14 |
|  | APPENDIX | 15 |

# viii

# CHAPTER 1

# INTRODUCTION

# Objective

# The objective of this project is to design and implement an Auction Management System that facilitates user registration, user authentication, auction creation, bidding, and displaying auction results. The system provides a user-friendly graphical interface and ensures robust database integration for secure and efficient data management

# Goals of the System:

# Allow users to register and log in.

# Enable auctioneers to create auctions.

# Allow bidders to place bids on items.

# Maintain the integrity of bids by validating input and updating the highest bid dynamically.

# Display auction results with details of winners and final bid amounts

# Overview

# The Auction Management System is a Java-based desktop application developed using Swing for the GUI and JDBC for database interaction. It integrates core Java programming concepts with a MySQL database to manage auctions efficiently.

# Key Features:

# 1.User Registration and Authentication:

# Secure user registration with validation.

# Login functionality to authenticate users

# 2.Auction Creation:

# Auctioneers can create auctions with details such as item name, starting price, and end time. 1

# 3.Bidding Process:

# Users can place bids on active auctions.

# Only valid bids (higher than the current highest bid) are accepted.

# 4.View Auctions and Results:

# Displays a list of active auctions.

# Shows final auction results, including the winning bidder and highest bid.

# 5.Database-Driven Application:

# All user and auction data are stored in and retrieved from a MySQL database

**1.3 JavaProgramming Concepts**

**1.Core Java Concepts:**

* OOP: Encapsulation within the AuctionManagementSystem class and modular design for functionality (e.g., registerUser, loginUser).
* Exception Handling: Manages errors like invalid input or database issues.

**2.GUI Development with Swing:**

* Components like JFrame, JPanel, JButton, and JTable create an interactive UI.
* Layouts (GridLayout, BorderLayout) and action listeners handle events.

**3.Database Connectivity (JDBC):**

* Uses DriverManager for MySQL connections and PreparedStatement for secure CRUD operations.
* ResultSet processes database query result

2

**4.Data Processing:**

* Validates inputs, parses data types, and formats dates with SimpleDateFormat.

**5.Dynamic Table Management:**

* Utilizes DefaultTableModel for updating auction data dynamically.

**6.Multi-Threading Potential:**

* Can be extended for real-time auction updates

3

**CHAPTER 2**

**PROJECT METHODOLOGY**

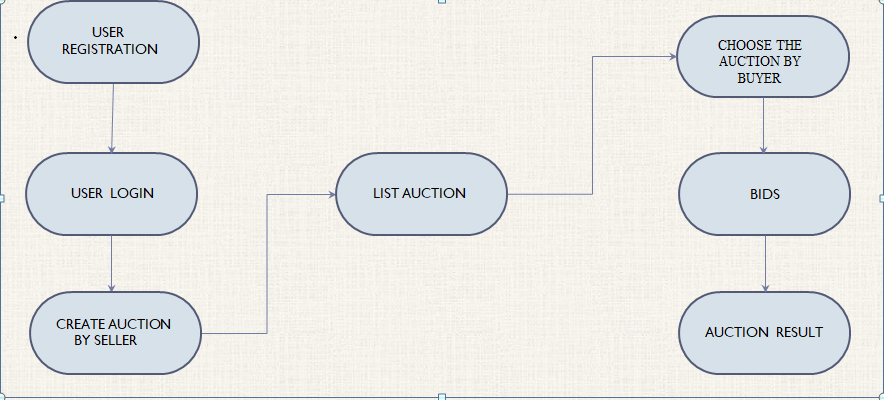
**2.1Proposed Work**

* **User Registration:** Users can register by providing a unique username and password. This information is securely stored, and upon successful registration, users are added to the system's list of registered users. Each registered user will be able to log in and participate in auctions either as a buyer or seller.
* **User Login:** Registered users can log in using their credentials. Upon successful login, they are granted access to create new auctions (if they are sellers) or to participate in ongoing auctions as buyers by placing bids on items.
* **Create Auction by Seller:** Once logged in, sellers can create an auction for an item by providing its name, a starting price, and a specific duration (end time). The auction is then made available for all users to view and bid on. This functionality ensures that sellers can seamlessly manage their items and set fair starting conditions for each auction.
* **List Auctions:** The system lists all ongoing auctions, providing details like the item name, highest bid placed so far, the seller’s username, and the remaining time for the auction. This ensures transparency and easy navigation for potential bidders.
* **Choose Auction by Buyer:** Buyers can select a specific auction to bid on by entering the item name. They are then prompted to enter the bid amount. The system verifies if the bid is higher than the current highest bid, allowing only valid bids to be placed.

4

* **Bids:** Buyers can place bids on items of interest. Each bid is compared to the current highest bid. If it surpasses the highest bid, it becomes the new leading bid. Buyers can continue to bid until the auction ends, ensuring competitive and fair bidding.
* **Auction Result:** When the auction end time is reached, the system checks for the highest bid. If a valid bid exists, the auction is concluded, and the system announces the winner (the user who placed the highest bid). If no bids were placed, the auction ends without a winner. The results of the auction, including the winning bidder, the bid amount, and the seller, are then displayed to ensure transparency.

**2.2 Block Diagram**



5

**CHAPTER 3**

**MODULE DESCRIPTION**

## 3.1 Module 1 :User Registration Module

## Allows users to register by providing a username and password.

## Validates input and stores details in the users table.

## 3.2 Module 2 : User Login Module

## Authenticates users using username and password.

## Establishes a session for valid users by querying the users table.

## 3.3 Module 3: Auction Creation Module

## Enables users to create auctions with item details (name, starting price, duration).

## Stores auction data in the auctions table.

## 3.4Module 4 : Auction Listing Module

## Displays active auctions, showing item name, highest bid, seller, and time remaining.

## Filters expired auctions via auctions table queries.

## 3.5Module 5 : Bid Placement Module

## Allows users to bid on active auctions.

## 6

## Validates bids, inserts records into the bids table, and updates auctions table with new highest bid.

## 3.6Module 6 : Result Module

## The Result Module displays completed auction outcomes, including auction ID, item name, starting price, highest bid, and winner details

## 

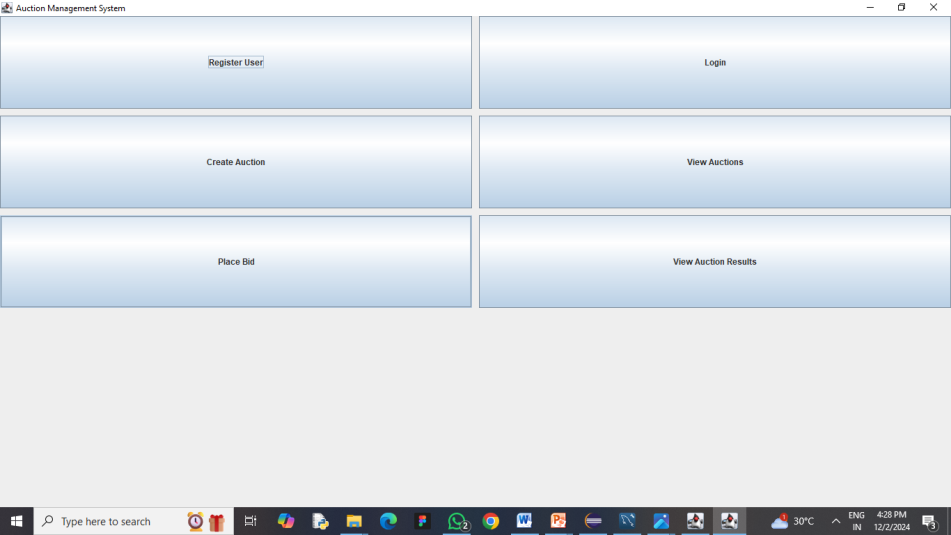
## 7

**CHAPTER 4**

**RESULTS AND DISCUSSION**

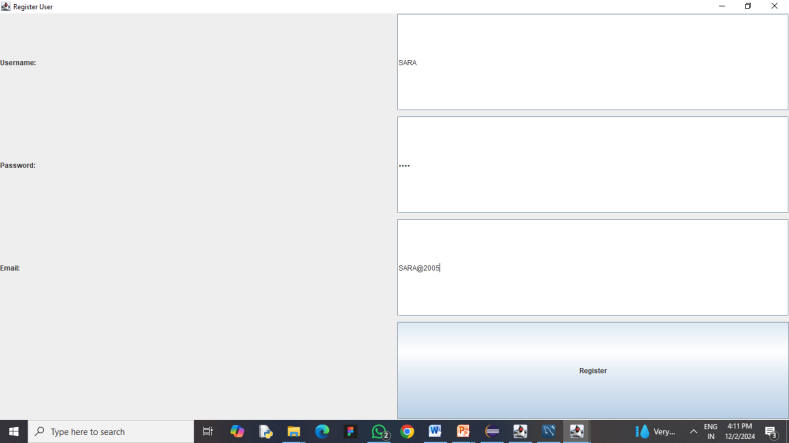
**1. MAIN MENU:**

* The main menu provides options for users to navigate through various functionalities, such as registering, logging in, creating auctions, viewing auctions, placing bids, and checking auction results



**2. REGISTRATION PAGE:**

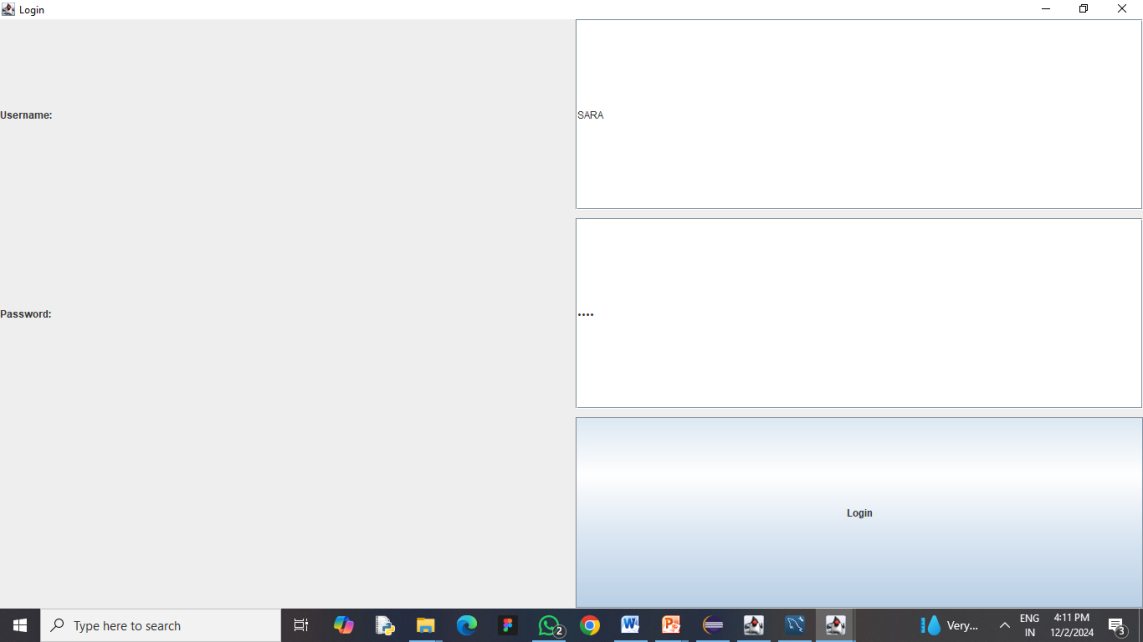
* Users can register by providing their username, password, and email.
* Inputs are validated and securely stored in the User Registration MySQL table.



8

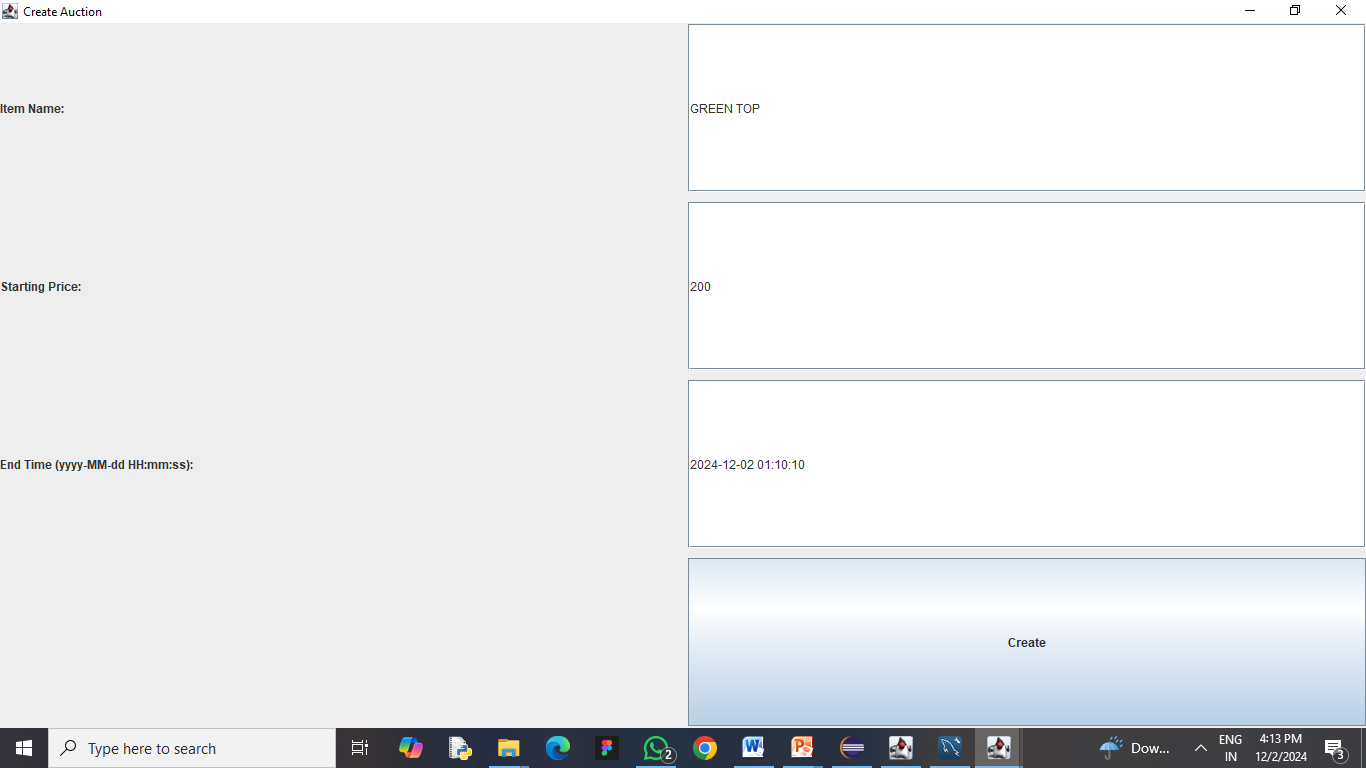
**3. LOGIN PAGE:**

* Enables users to log in with their credentials.
* Validates login details against the User Registration table.



**4.CREATE AUCTION:**

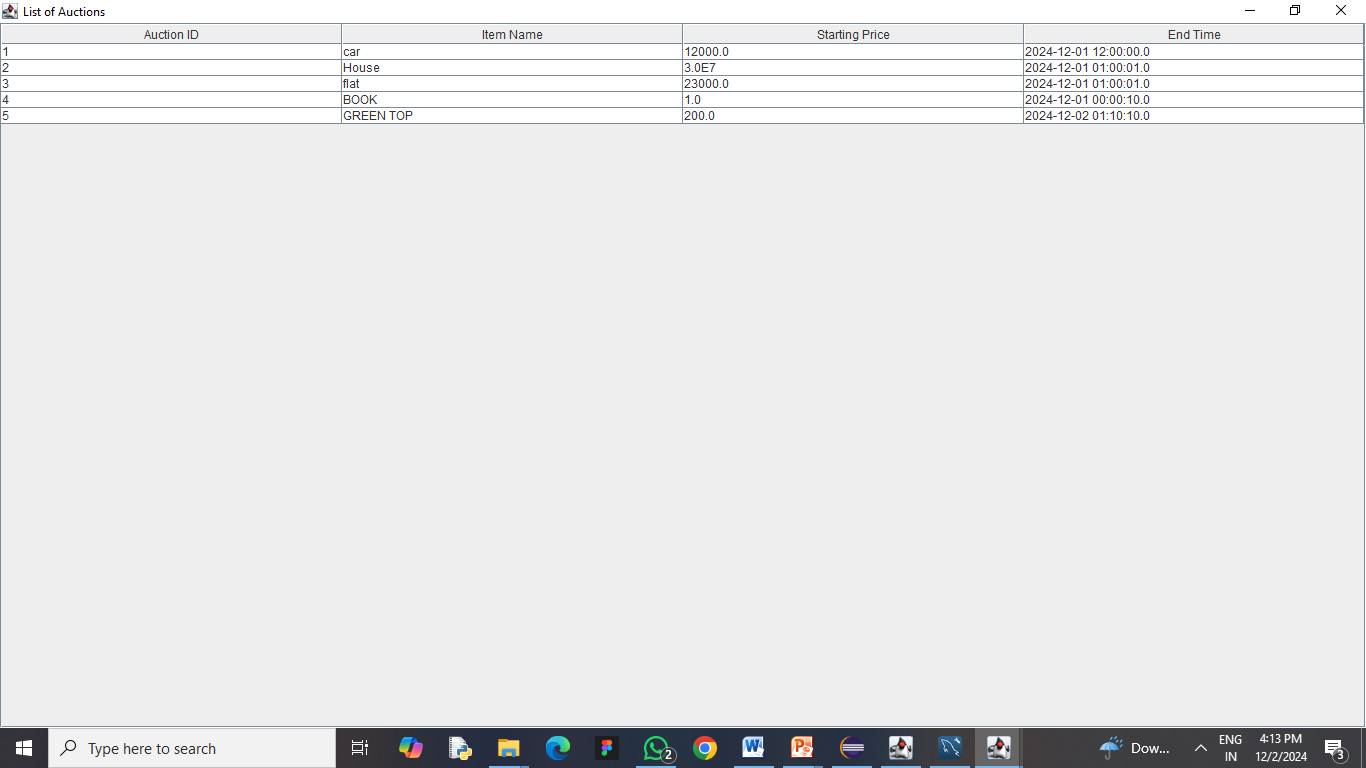
* Allows users to create new auctions by entering item name, starting price, and auction end time.
* Data is stored in the Create Auction MySQL table

­­­­

9

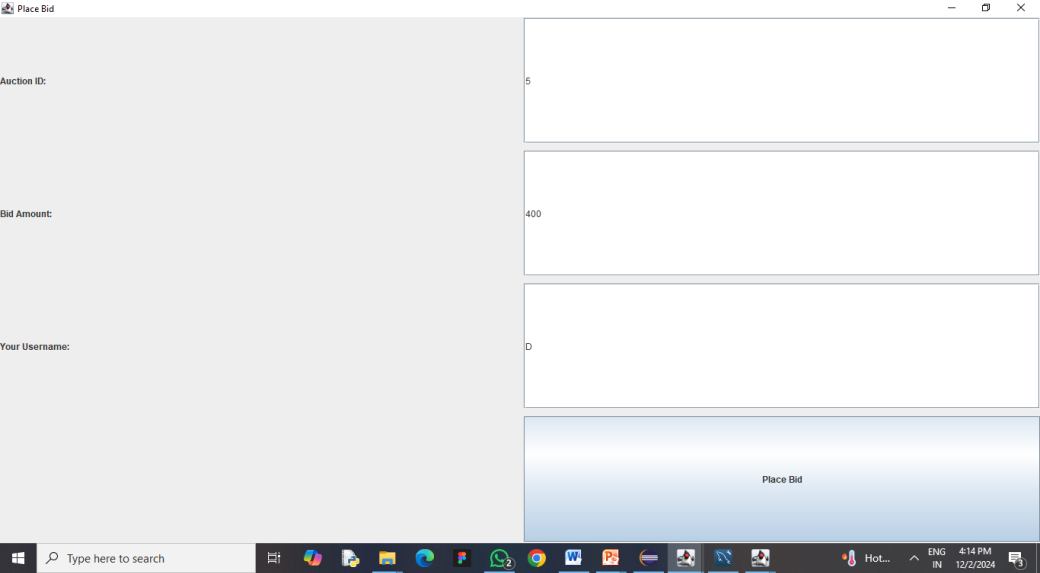
**5.LIST OF AUCTIONS:**

* Displays all active auctions in a dynamic table format using data from the Create Auction table.
* Details include Auction ID, Item Name, Starting Price, and End Time



**6.BID PLACED:**

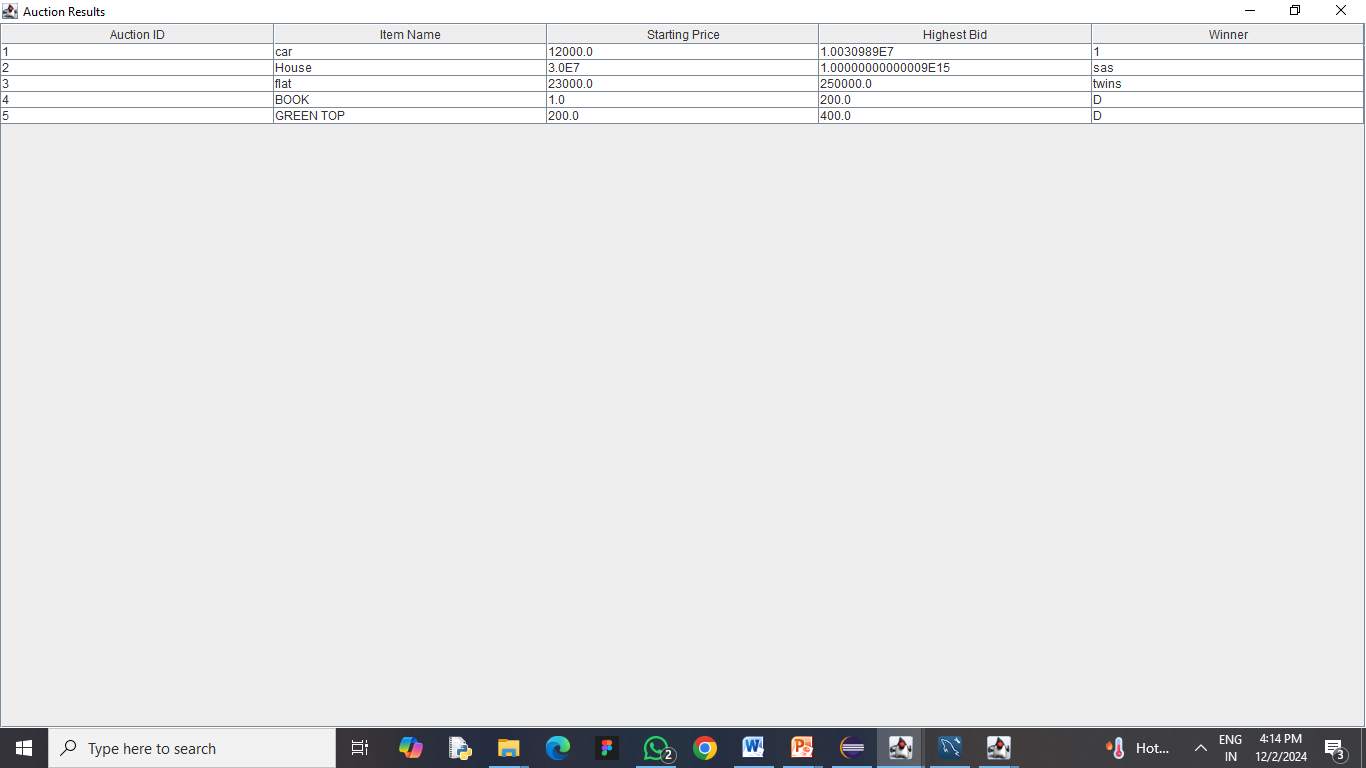
* Users can place bids on active auctions.
* Validates that the bid is higher than the current highest bid and updates the Bid Placed MySQL table



10

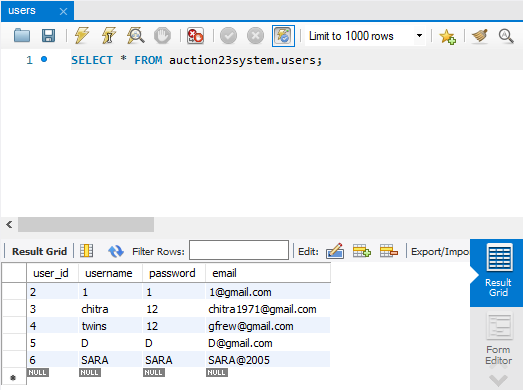
**7. RESULT OF THE AUCTION:**

* Displays the outcomes of completed auctions, including the highest bid and winner information.
* Results are dynamically fetched from the Create Auction table and joined with user details from the User Registration table.



**8. USER REGISTERATION MYSQL TABLE:**

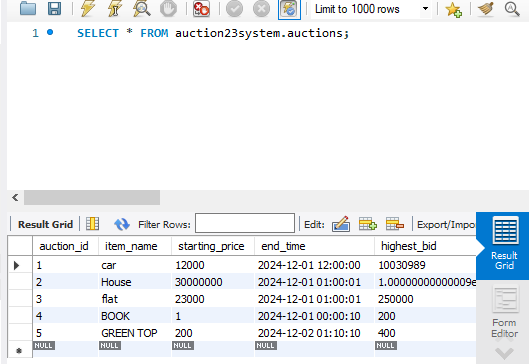
* Stores user details like user\_id, username, password, and email.
* Ensures unique usernames and securely saves credentials.



**11**

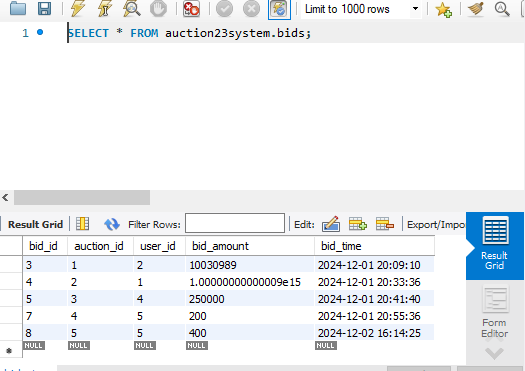
**9.CREATE AUCTION MYSQL TABLE:**

* Tracks auction details such as auction\_id, item\_name, starting\_price, end\_time, and winner\_id.
* Links winners to the User Registration table via winner\_id



**10.BID PLACED MYSQL TABLE:**

* Records bids with bid\_id, auction\_id, user\_id, and bid\_amount.
* Links to Create Auction and User Registration tables for bid tracking.



12

**CHAPTER 5**

**CONCLUSION**

The Auction Management System successfully integrates key Java programming concepts and database operations to provide a robust platform for managing auctions. With its user-friendly interface, secure database connectivity, and modular design, the system achieves its goals of facilitating user registration, auction creation, bidding, and result display.

This project demonstrates the effective use of Java Swing for GUI development and JDBC for data management, ensuring both functionality and security. The dynamic and interactive features, such as real-time bid updates and detailed result displays, enhance the user experience.

Future enhancements, like real-time notifications and additional analytics, can further improve the system's efficiency and scalability, making it adaptable to larger-scale implementations. Overall, the project provides a solid foundation for auction management solutions.

­

13

**REFERENCES:**

1. **Java** **Documentation:**

* Official Oracle Java SE Documentation: https://docs.oracle.com/javase/
* Java Swing Documentation: https://docs.oracle.com/javase/tutorial/uiswing/

1. **Database Management:**

* MySQL Official Documentation: https://dev.mysql.com/doc/
* JDBC Guide: https://docs.oracle.com/javase/tutorial/jdbc/

1. **Programming Resources:**

* Stack Overflow: Solutions for common programming issues and best practices.
* GeeksforGeeks: Tutorials on Java and Swing programming concepts.

14

**APPENDIX**

**(Coding)**

**JAVA CODE:**

package javaconnectivity;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

public class AuctionManagementSystem {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/auction23system";

private static final String DB\_USER = "root";

private static final String DB\_PASSWORD = "sasmitha@2005";

private Connection connection;

public AuctionManagementSystem() {

try {

connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

System.out.println("Connected to the database.");

showMainMenu();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(AuctionManagementSystem::new);

}

private void showMainMenu() {

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

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(5, 1, 10, 10));

JButton registerButton = new JButton("Register User");

JButton loginButton = new JButton("Login");

JButton createAuctionButton = new JButton("Create Auction");

JButton viewAuctionsButton = new JButton("View Auctions");

JButton placeBidButton = new JButton("Place Bid");

JButton viewResultsButton = new JButton("View Auction Results");

registerButton.addActionListener(e -> showRegisterForm(frame));

loginButton.addActionListener(e -> showLoginForm(frame));

createAuctionButton.addActionListener(e -> showCreateAuctionForm(frame));

16

viewAuctionsButton.addActionListener(e -> showAuctionList(frame));

placeBidButton.addActionListener(e -> showPlaceBidForm(frame));

viewResultsButton.addActionListener(e -> showAuctionResults(frame));

panel.add(registerButton);

panel.add(loginButton);

panel.add(createAuctionButton);

panel.add(viewAuctionsButton);

panel.add(placeBidButton);

panel.add(viewResultsButton);

frame.add(panel);

frame.setVisible(true);

}

private void showRegisterForm(JFrame parentFrame) {

JFrame frame = new JFrame("Register User");

frame.setSize(300, 250);

JPanel panel = new JPanel(new GridLayout(4, 2, 10, 10));

JLabel usernameLabel = new JLabel("Username:");

JTextField usernameField = new JTextField();

JLabel passwordLabel = new JLabel("Password:");

JPasswordField passwordField = new JPasswordField();

JLabel emailLabel = new JLabel("Email:");

JTextField emailField = new JTextField();

JButton registerButton = new JButton("Register");

17

registerButton.addActionListener(e -> {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

String email = emailField.getText();

registerUser(username, password, email);

frame.dispose();

});

panel.add(usernameLabel);

panel.add(usernameField);

panel.add(passwordLabel);

panel.add(passwordField);

panel.add(emailLabel);

panel.add(emailField);

panel.add(new JLabel()); // Empty space

panel.add(registerButton);

frame.add(panel);

frame.setVisible(true);

}

private void registerUser(String username, String password, String email) {

try {

String sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, username);

18

statement.setString(2, password);

statement.setString(3, email);

statement.executeUpdate();

JOptionPane.showMessageDialog(null, "User registered successfully!");

} catch (SQLException e) {

e.printStackTrace();

}

}

private void showLoginForm(JFrame parentFrame) {

JFrame frame = new JFrame("Login");

frame.setSize(300, 200);

JPanel panel = new JPanel(new GridLayout(3, 2, 10, 10));

JLabel usernameLabel = new JLabel("Username:");

JTextField usernameField = new JTextField();

JLabel passwordLabel = new JLabel("Password:");

JPasswordField passwordField = new JPasswordField();

JButton loginButton = new JButton("Login");

loginButton.addActionListener(e -> {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

if (loginUser(username, password)) {

JOptionPane.showMessageDialog(null, "Login successful!");

frame.dispose();

} else {

19

JOptionPane.showMessageDialog(null, "Invalid credentials.");

}

});

panel.add(usernameLabel);

panel.add(usernameField);

panel.add(passwordLabel);

panel.add(passwordField);

panel.add(new JLabel()); // Empty space

panel.add(loginButton);

frame.add(panel);

frame.setVisible(true);

}

private boolean loginUser(String username, String password) {

try {

String sql = "SELECT \* FROM users WHERE username = ? AND password = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, username);

statement.setString(2, password);

return statement.executeQuery().next();

} catch (SQLException e) {

e.printStackTrace();

}

return false;

} 20

private void showCreateAuctionForm(JFrame parentFrame) {

JFrame frame = new JFrame("Create Auction");

frame.setSize(300, 250);

JPanel panel = new JPanel(new GridLayout(4, 2, 10, 10));

JLabel itemNameLabel = new JLabel("Item Name:");

JTextField itemNameField = new JTextField();

JLabel startingPriceLabel = new JLabel("Starting Price:");

JTextField startingPriceField = new JTextField();

JLabel endTimeLabel = new JLabel("End Time (yyyy-MM-dd HH:mm:ss):");

JTextField endTimeField = new JTextField();

JButton createButton = new JButton("Create");

createButton.addActionListener(e -> {

String itemName = itemNameField.getText();

double startingPrice = Double.parseDouble(startingPriceField.getText());

String endTimeStr = endTimeField.getText();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date endTime = null;

try {

endTime = sdf.parse(endTimeStr);

} catch (Exception ex) {

ex.printStackTrace();

}

createAuction(itemName, startingPrice, new

21

Timestamp(endTime.getTime()));

frame.dispose();

});

panel.add(itemNameLabel);

panel.add(itemNameField);

panel.add(startingPriceLabel);

panel.add(startingPriceField);

panel.add(endTimeLabel);

panel.add(endTimeField);

panel.add(new JLabel()); // Empty space

panel.add(createButton);

frame.add(panel);

frame.setVisible(true);

}

private void createAuction(String itemName, double startingPrice, Timestamp endTime) {

try {

String sql = "INSERT INTO auctions (item\_name, starting\_price, end\_time) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, itemName);

statement.setDouble(2, startingPrice);

statement.setTimestamp(3, endTime);

statement.executeUpdate();

22

JOptionPane.showMessageDialog(null, "Auction created successfully!");

} catch (SQLException e) {

e.printStackTrace();

}

}

private void showPlaceBidForm(JFrame parentFrame) {

JFrame frame = new JFrame("Place Bid");

frame.setSize(300, 250);

JPanel panel = new JPanel(new GridLayout(4, 2, 10, 10));

JLabel auctionIdLabel = new JLabel("Auction ID:");

JTextField auctionIdField = new JTextField();

JLabel bidAmountLabel = new JLabel("Bid Amount:");

JTextField bidAmountField = new JTextField();

JLabel usernameLabel = new JLabel("Your Username:");

JTextField usernameField = new JTextField();

JButton bidButton = new JButton("Place Bid");

bidButton.addActionListener(e -> {

int auctionId = Integer.parseInt(auctionIdField.getText());

double bidAmount = Double.parseDouble(bidAmountField.getText());

String username = usernameField.getText();

placeBid(username, auctionId, bidAmount);

frame.dispose();

});

23

panel.add(auctionIdLabel);

panel.add(auctionIdField);

panel.add(bidAmountLabel);

panel.add(bidAmountField);

panel.add(usernameLabel);

panel.add(usernameField);

panel.add(new JLabel()); // Empty space

panel.add(bidButton);

frame.add(panel);

frame.setVisible(true);

}

private void showAuctionList(JFrame parentFrame) {

JFrame frame = new JFrame("List of Auctions");

frame.setSize(600, 400);

String[] columnNames = {"Auction ID", "Item Name", "Starting Price", "End Time"};

DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);

try {

// Query to fetch auction details

String sql = "SELECT auction\_id, item\_name, starting\_price, end\_time FROM auctions";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

24

while (resultSet.next()) {

int auctionId = resultSet.getInt("auction\_id");

String itemName = resultSet.getString("item\_name");

double startingPrice = resultSet.getDouble("starting\_price");

Timestamp endTime = resultSet.getTimestamp("end\_time");

// Adding rows to the table model

tableModel.addRow(new Object[]{auctionId, itemName, startingPrice, endTime});

}

} catch (SQLException e) {

e.printStackTrace();

}

// Create JTable to display auction list

JTable table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

frame.add(scrollPane);

frame.setVisible(true);

}

private void placeBid(String username, int auctionId, double bidAmount) {

try {

// Check if the bid is higher than the current highest bid

25

String checkBidSql = "SELECT highest\_bid FROM auctions WHERE

auction\_id = ?";

PreparedStatement checkBidStatement = connection.prepareStatement(checkBidSql);

checkBidStatement.setInt(1, auctionId);

ResultSet resultSet = checkBidStatement.executeQuery();

if (resultSet.next()) {

double highestBid = resultSet.getDouble("highest\_bid");

if (bidAmount <= highestBid) {

JOptionPane.showMessageDialog(null, "Bid must be higher than the current highest bid.");

return;

}

}

// Place the bid

String getUserIdSql = "SELECT user\_id FROM users WHERE username = ?";

PreparedStatement getUserIdStatement = connection.prepareStatement(getUserIdSql);

getUserIdStatement.setString(1, username);

ResultSet userResult = getUserIdStatement.executeQuery();

if (userResult.next()) {

int userId = userResult.getInt("user\_id");

26

String insertBidSql = "INSERT INTO bids (auction\_id, user\_id,

bid\_amount) VALUES (?, ?, ?)";

PreparedStatement insertBidStatement = connection.prepareStatement(insertBidSql);

insertBidStatement.setInt(1, auctionId);

insertBidStatement.setInt(2, userId);

insertBidStatement.setDouble(3, bidAmount);

insertBidStatement.executeUpdate();

// Update the highest bid for the auction

String updateAuctionSql = "UPDATE auctions SET highest\_bid = ?, winner\_id = ? WHERE auction\_id = ?";

PreparedStatement updateAuctionStatement = connection.prepareStatement(updateAuctionSql);

updateAuctionStatement.setDouble(1, bidAmount);

updateAuctionStatement.setInt(2, userId);

updateAuctionStatement.setInt(3, auctionId);

updateAuctionStatement.executeUpdate();

JOptionPane.showMessageDialog(null, "Bid placed successfully!");

} else {

JOptionPane.showMessageDialog(null, "User not found.");

}

} catch (SQLException e) {

e.printStackTrace();

}

} 27

private void showAuctionResults(JFrame parentFrame) {

JFrame frame = new JFrame("Auction Results");

frame.setSize(600, 400);

JPanel panel = new JPanel(new BorderLayout());

String[] columnNames = {"Auction ID", "Item Name", "Starting Price", "Highest Bid", "Winner"};

DefaultTableModel model = new DefaultTableModel(columnNames, 0);

JTable table = new JTable(model);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

// Fetch auction data and display it

try {

String sql = "SELECT a.auction\_id, a.item\_name, a.starting\_price, a.highest\_bid, u.username AS winner " +

"FROM auctions a LEFT JOIN users u ON a.winner\_id = u.user\_id";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

int auctionId = resultSet.getInt("auction\_id");

String itemName = resultSet.getString("item\_name");

double startingPrice = resultSet.getDouble("starting\_price");

double highestBid = resultSet.getDouble("highest\_bid");

28

String winner = resultSet.getString("winner");

model.addRow(new Object[]{auctionId, itemName, startingPrice, highestBid, winner});

}

} catch (SQLException e) {

e.printStackTrace();

}

frame.add(panel);

frame.setVisible(true);

}

}

**MYSQL CODE:**

CREATE DATABASE auction23system;

USE auction23system;

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL

);

CREATE TABLE auctions (

auction\_id INT AUTO\_INCREMENT PRIMARY KEY,

item\_name VARCHAR(100) NOT NULL,

starting\_price DOUBLE NOT NULL,

end\_time TIMESTAMP NOT NULL,

29

highest\_bid DOUBLE DEFAULT 0,

winner\_id INT,

FOREIGN KEY (winner\_id) REFERENCES users(user\_id)

);

CREATE TABLE bids (

bid\_id INT AUTO\_INCREMENT PRIMARY KEY,

auction\_id INT,

user\_id INT,

bid\_amount DOUBLE,

bid\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (auction\_id) REFERENCES auctions(auction\_id),

FOREIGN KEY (user\_id) REFERENCES users(user\_id));

30