ONLINE AUCTION SYSTEM DOCUMENTATION

Gleb Tutubalin (C00290944) , Aaron Brooke (C00296625)

YEAR 2 Project

Contents

1. Description

The Online Auction System is a Java-based application that allows users to participate in online auctions. Traditional auctions face challenges including geographic limitations and fixed schedules, which this system addresses by creating an online auction environment accessible from anywhere nationwide.

The system provides a platform for users to: - Register accounts and authenticate securely - Create auctions for items they want to sell - Browse available auctions - Place bids on items in real-time - Monitor auction progress - Complete transactions through secure payment processing - Submit reports for issues or concerns

Key features include real-time bid updates, automatic auction closing, image upload capabilities for auction items, and a consistent user interface that guides users through the auction process from registration to payment completion.

2. Requirements

Functional Requirements

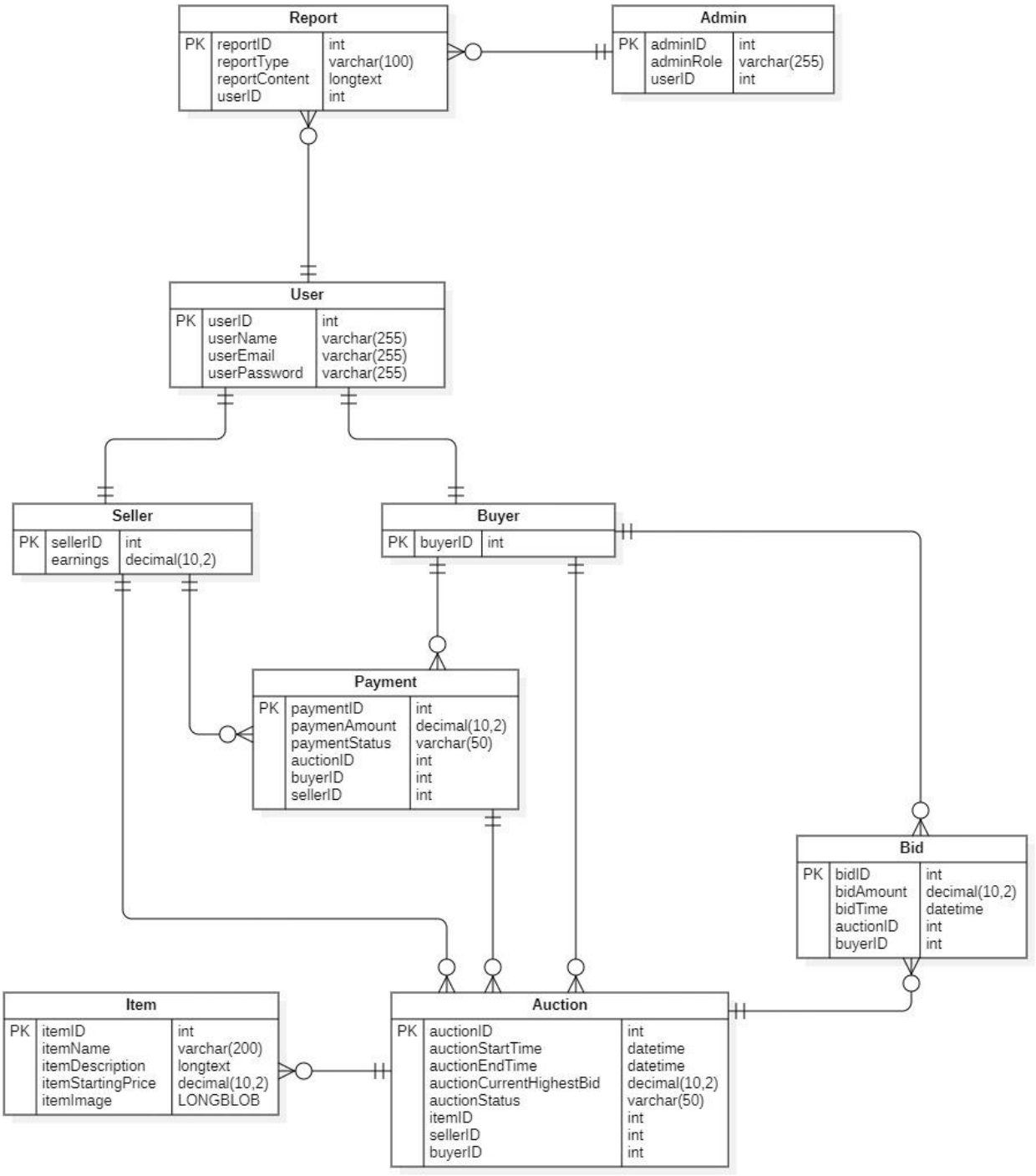
1. **User Management**
   - Users must be able to register accounts with name, email, and password
   - Users must be able to log in using email and password
   - Users must be able to view their profile information
2. **Auction Management**
   - Sellers must be able to create new auctions with detailed item information
   - Sellers must be able to upload images for auction items
   - Sellers must be able to set auction parameters (starting price, end time)
   - Users must be able to browse active auctions
3. **Bidding System**
   - Buyers must be able to place bids on active auctions
   - The system must validate bids against current highest bid
   - The system must update auction status in real-time
4. **Payment Processing**
   - Buyers must be able to view and process pending payments
   - The system must automatically create payment records when auctions close
   - The system must track payment status changes
5. **Reporting System**
   - Users must be able to submit reports with title and description
   - The system must store report information

Non-Functional Requirements

1. **Usability**
   - The interface must be intuitive with consistent navigation
   - Error messages must clearly indicate validation issues
2. **Performance**
   - The system must refresh auction data regularly (at least every 5 seconds)
   - Image loading must be optimized for performance
3. **Reliability**
   - The system must properly handle database connections
   - The system must recover gracefully from errors
4. **Maintainability**
   - The code must follow a modular architecture (MVC pattern)
   - Common functionality must be abstracted to reduce duplication

3. Database Structure

ER Diagram

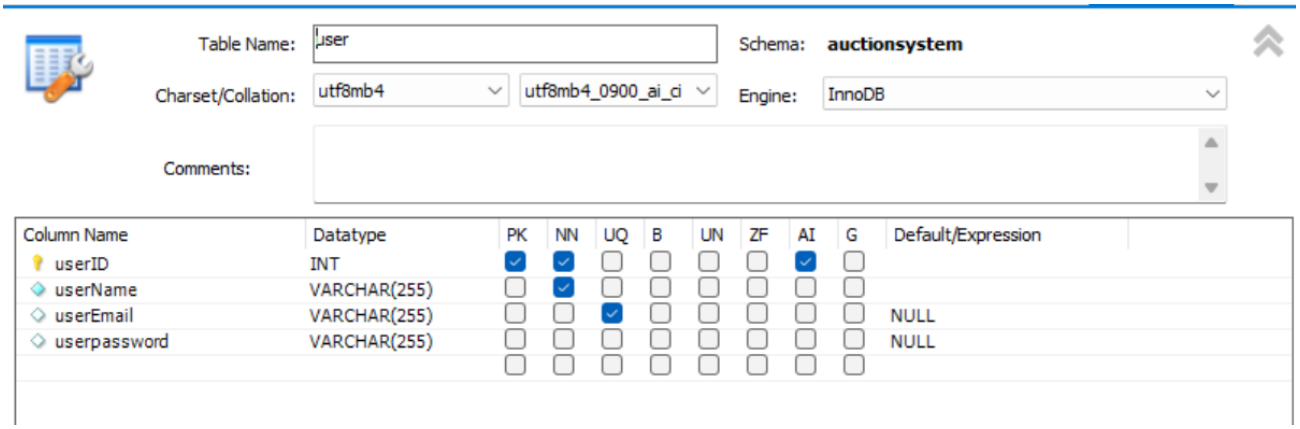


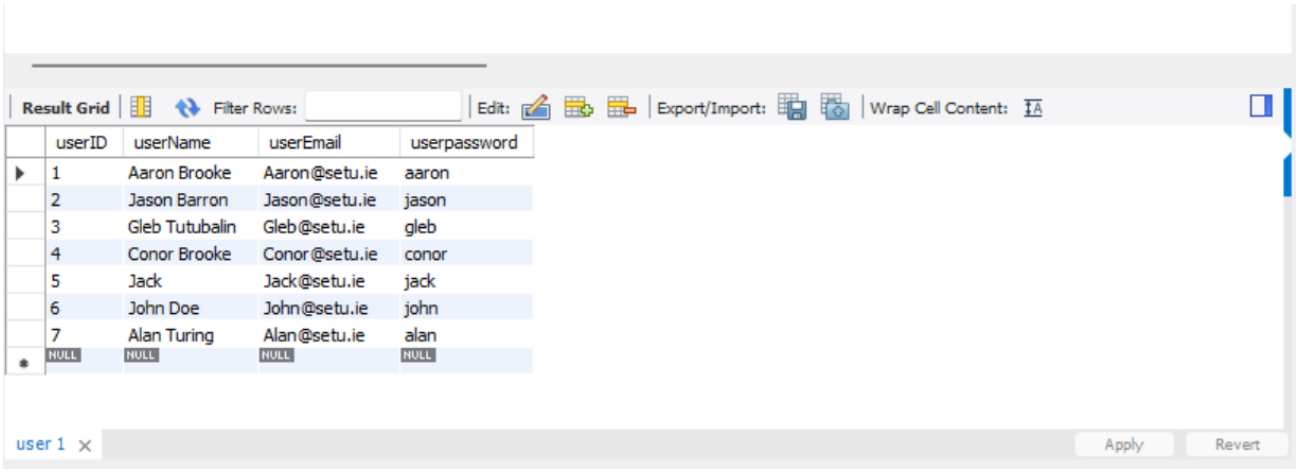
The database consists of the following key entities:

- **User**: Stores account information
- **Admin**: Controls user management, user reports/issues, and views Auctions.
- **Buyer**: Browses through a list of auctions and places bids on items.
- **Seller**: Creates auctions online and can bid on items.
- **Item**: Stores auction item details including images
- **Auction**: Manages the auction lifecycle and status
- **Bid**: Tracks all bids placed at auctions
- **Payment**: Records, payment information and status
- **Report**: Stores user-submitted reports

Database Tables (Structure & Data)

User Table Structure





Admin Table Structure

Table Name: admin Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
adminID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
adminPrivileges	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
userID	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

adminID	adminPrivileges	userID
1	admin	2
2	Super Admin	5
NULL	NULL	NULL

Item Table Structure

Table Name: item Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
itemID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
itemName	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
itemDescription	TEXT	☐	☑	☐	☐	☐	☐	☐	☐	
itemStartingPrice	DECIMAL(10,2)	☐	☑	☐	☐	☐	☐	☐	☐	
itemImage	LONGBLOB	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

itemID	itemName	itemDescription	itemStartingPrice	itemImage
1	Patek Philippe Grandmaster Chime 6300A	Rare Watch	5000000.00	BLOB
2	Table	Wooden table	300.00	BLOB
3	Gaming Laptop	MSI GF 75 Thin 10SER Gaming Laptop	400.00	BLOB
4	Watch	Patek Philippe Grandmaster Chime 6300A	5100000.00	BLOB
5	Bombe	Bombe from WarII	5000000.00	BLOB
NULL	NULL	NULL	NULL	NULL

item 1 | Apply | Revert

Auction Table Structure

Table Name: auction Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
auctionID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
auctionStartTime	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
auctionEndTime	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
auctionCurrentHighestBid	DECIMAL(10,2)	☐	☐	☐	☐	☐	☐	☐	☐	'0.00'
auctionStatus	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐	☐	
itemID	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
sellerID	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

auctionID	auctionStartTime	auctionEndTime	auctionCurrentHighestBid	auctionStatus	itemID	sellerID	buyerID
1	2025-04-11 18:09:23	2025-04-11 19:45:00	0.00	STARTED	1	1	0
2	2025-04-11 18:09:57	2025-04-12 18:09:37	500.00	STARTED	2	1	4
3	2025-04-11 18:18:57	2025-04-12 18:14:24	550.00	STARTED	3	3	7
4	2025-04-11 18:20:47	2025-04-12 18:19:35	6500000.00	STARTED	4	3	1
5	2025-04-11 18:35:53	2025-04-11 18:40:00	7000000.00	STARTED	5	7	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

auction 1 × Apply Revert

Bid Table Structure

Table Name: bid Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
bidID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
bidAmount	DECIMAL(10,2)	☐	☑	☐	☐	☐	☐	☐	☐	
bidTime	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
auctionID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
buyerID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

bidID	bidAmount	bidTime	auctionID	buyerID
1	350.00	2025-04-11 18:13:44	2	1
2	400.00	2025-04-11 18:19:14	2	3
3	5700000.00	2025-04-11 18:21:09	4	3
4	550.00	2025-04-11 18:36:07	3	7
5	6000000.00	2025-04-11 18:36:21	4	7
6	6000000.00	2025-04-11 18:37:58	5	1
7	6500000.00	2025-04-11 18:38:23	4	1
8	7000000.00	2025-04-11 18:39:11	5	4
9	500.00	2025-04-11 18:39:22	2	4
NULL	NULL	NULL	NULL	NULL

bid 1 | Apply | Revert

Payment Table Structure

Table Name: payment Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
paymentID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
paymentAmount	DECIMAL(10,2)	☐	☑	☐	☐	☐	☐	☐	☐	
paymentStatus	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐	☐	
auctionID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
buyerID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
sellerID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

paymentID	paymentAmount	paymentStatus	auctionID	buyerID	sellerID
1	5000000.00	PENDING	8	1	1
NULL	NULL	NULL	NULL	NULL	NULL

payment 1 × Apply Revert

Output

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

paymentID	paymentAmount	paymentStatus	auctionID	buyerID	sellerID
1	5000000.00	COMPLETED	8	1	1
NULL	NULL	NULL	NULL	NULL	NULL

payment 1 × Apply Revert

Report Table Structure

Table Name: report Schema: **auctionsystem**

Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

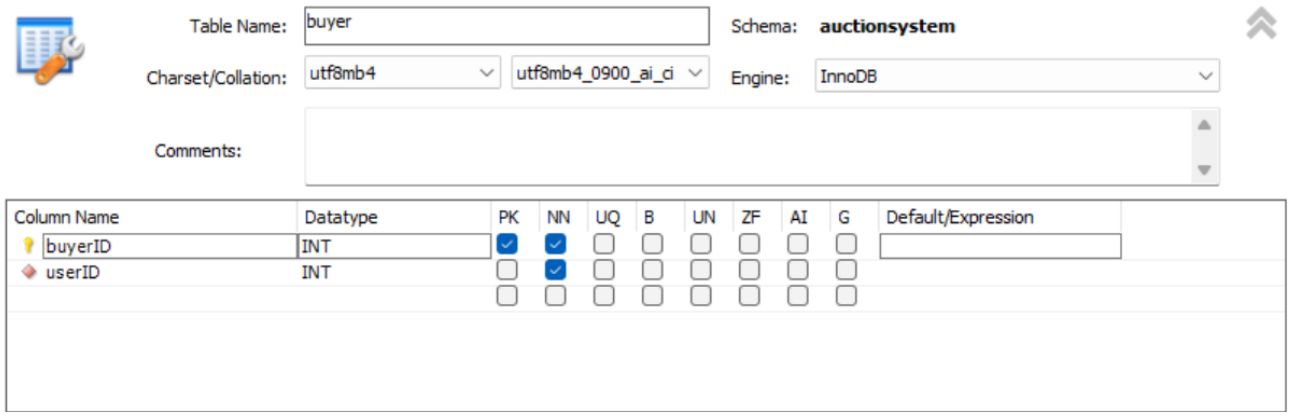
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
reportID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
reportType	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
reportContent	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
userID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

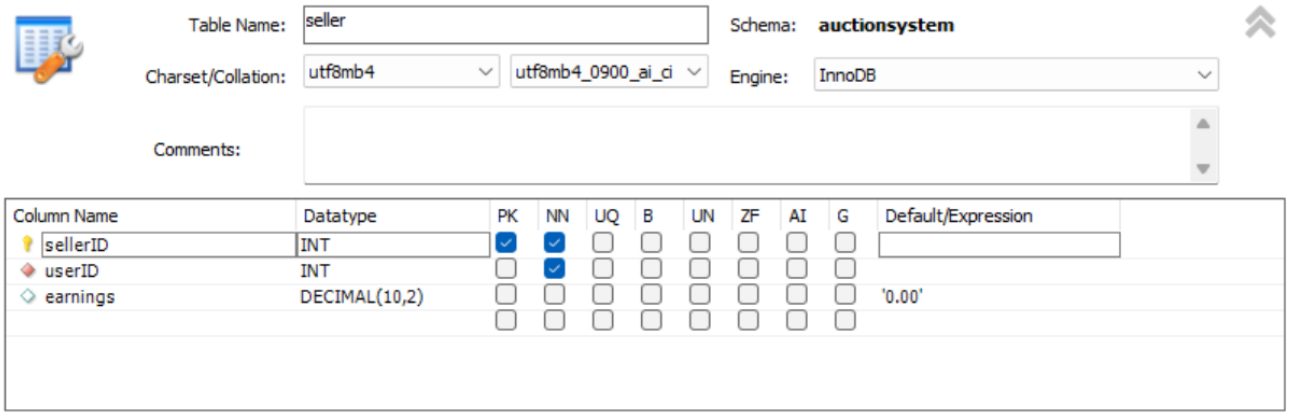
reportID	reportType	reportContent	userID
1	Patek Philippe Grandmaster Chime 6300A	No image for item listed on auction?	1
2	No Watch Image	Image not loading for watch?	7
NULL	NULL	NULL	NULL

report 1 × Apply Revert

Buyer Table Structure



Seller Table Structure



4. Interesting Source Code Snippets

1. BaseGUI Abstract Class

The BaseGUI class implements a reusable framework for creating consistent GUI screens. This is a great example of using abstraction to reduce code duplication across UI classes:

```
public abstract class BaseGUI extends JFrame {
    private final int userID;
    private JLabel dateTimeLabel;
    private Timer dateTimeTimer;
    private JLabel errorMessageLabel;

    public BaseGUI(int userID, String title, boolean useDispose) {
        this.userID = userID;

        // Set up the frame
        setTitle(title);
        setSize(900, 700);
        setDefaultCloseOperation(useDispose ? JFrame.DISPOSE_ON_CLOSE :
JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Set background color
        getContentPane().setBackground(new Color(240, 240, 240));

        // Set up common components
        setupHeader(title);
        setupMenuBar();
        setupErrorLabel();

        // Add window listener for cleanup
        addWindowListener(new java.awt.event.WindowAdapter() {
            @Override
            public void windowClosing(java.awt.event.WindowEvent windowEvent) {
                cleanup();
            }
        });
    }

    // Methods for setting up common UI elements and handling navigation

    protected void cleanup() {
        if (dateTimeTimer != null) {
            dateTimeTimer.stop();
        }
    }
}
```

2. Real-time Auction Monitoring

This code snippet from `MonitorAuctionGUI` demonstrates how the system provides real-time updates of auction status using Swing timers:

```
private void startUpdateTimer() {
    updateTimer = new Timer(1000, e -> {
        try {
            // Refresh auction details
            auction = AuctionDAO.getAuctionByID(auction.getAuctionID());
            lblHighestBid.setText("Highest Bid: €" +
auction.getAuctionCurrentHighestBid());
            lblAuctionStatus.setText("Status: " + auction.getAuctionStatus());

            // Refresh bid history
            loadBidHistory();

            // Check if auction is closed
            if (auction.getAuctionStatus().equalsIgnoreCase("CLOSED") ||
                auction.getAuctionStatus().equalsIgnoreCase("FINISHED")) {
                updateTimer.stop();
                JOptionPane.showMessageDialog(this, "Auction has ended. Payment
will be processed.",
                    "Auction Closed", JOptionPane.INFORMATION_MESSAGE);

                // Remove the bid panel
                removeBidPanel();

                // Process payment
                processPayment();
            }
        } catch (Exception ex) {
            setErrorMessage("Error updating auction details: " +
ex.getMessage());
            ex.printStackTrace();
        }
    });
    updateTimer.start();
}
```

3. Database Connection Management

This snippet from `DBConnector` shows how database connections are managed to prevent resource leaks:

```
public class DBConnector {
    // DB connection configuration
    private static final String USER = "root";
    private static final String PASSWORD = "TeaAndHoney12!";
    private static final String URL =
```

```
"jdbc:mysql://localhost:3306/auctionsystem";

    private static Connection con;

    // Get a database connection
    public static Connection getConnection() throws SQLException {
        if (con == null || con.isClosed()) {
            con = DriverManager.getConnection(URL, USER, PASSWORD);
        }
        return con;
    }

    // Test connection method
    public static void main(String[] args) {
        // Test database connection using a try catch
        try {
            createConnection();
            if (con != null) {
                System.out.println("Successfully connected to the DB");
            }
        } catch (SQLException e) {
            System.err.println("Failed to connect to DB");
            e.printStackTrace();
        } finally {
            try {
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

4. Input Validation

This code from `CreateAuctionGUI` shows comprehensive validation for auction creation:

```
private boolean validateItemDetails() {
    String itemName = txtItemName.getText().trim();
    String itemDesc = txtItemDesc.getText().trim();
    String startPriceText = txtStartPrice.getText().trim();

    if (itemName.isEmpty()) {
        setErrorMessage("Item Name cannot be empty.");
        return false;
    }

    if (itemDesc.isEmpty()) {
        setErrorMessage("Item Description cannot be empty.");
        return false;
```

```
    }

    try {
        double startPrice = Double.parseDouble(startPriceText);
        if (startPrice <= 0) {
            setErrorMessage("Starting Price must be greater than 0.");
            return false;
        }
    } catch (NumberFormatException e) {
        setErrorMessage("Starting Price must be a valid number.");
        return false;
    }

    if (itemImage == null) {
        setErrorMessage("Please select an image for the item.");
        return false;
    }

    // Validate auction end time
    LocalDateTime endTime = getDateTimeFromSpinner(spinnerEnd);
    LocalDateTime currentTime = LocalDateTime.now();
    if (!endTime.isAfter(currentTime)) {
        setErrorMessage("Auction End Time must be later than the current
time.");
        return false;
    }

    clearErrorMessage(); // Clear error message
    return true;
}
```

5. Image Handling

This snippet demonstrates how the system handles image uploads for auction items:

```
private void chooseImage() {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    int result = fileChooser.showOpenDialog(this);

    if (result == JFileChooser.APPROVE_OPTION) {
        try {
            File selectedFile = fileChooser.getSelectedFile();
            // Check file size before loading
            if (selectedFile.length() > 10_000_000) { // 10MB limit
                setErrorMessage("Image too large (max 10MB)");
                imageStatusLabel.setText("Image too large (max 10MB)");
                imageStatusLabel.setForeground(Color.RED);
                return;
            }
```

```
            itemImage = Files.readAllBytes(selectedFile.toPath());
            imageStatusLabel.setText("Image selected: " +
selectedFile.getName());
            imageStatusLabel.setForeground(new Color(0, 128, 0)); // Dark green
        } catch (IOException e) {
            setErrorMessage("Failed to load image: " + e.getMessage());
            imageStatusLabel.setText("Failed to load image");
            imageStatusLabel.setForeground(Color.RED);
        }
    } else {
        imageStatusLabel.setText("No image selected");
        imageStatusLabel.setForeground(Color.RED);
    }
}
```

5. Tests

The system has been tested to ensure it meets all requirements. The following test cases were executed:

User Management Tests

Test Case	Description	Steps	Expected Result	Status
TC001	Register with valid data	1. Open registration page 2. Enter name, email, password 3. Submit form	User account is created successfully	PASS
TC002	Register with invalid data	1. Open registration page 2. Leave fields empty 3. Submit form	Error messages are displayed	PASS
TC003	Login with valid credentials	1. Open login page 2. Enter valid email and password 3. Submit form	User is logged in and redirected to main menu	PASS
TC004	Login with invalid credentials	1. Open login page 2. Enter an invalid email or password 3. Submit form	Error message is displayed	PASS

Auction Management Tests

Test Case	Description	Steps	Expected Result	Status
TC005	Create auction with valid data	1. Navigate to Create Auction 2. Fill in all fields 3. Upload image 4. Submit form	Auction is created successfully	PASS
TC006	Create auction with invalid data	1. Navigate to Create Auction 2. Leave required fields empty 3. Submit form	Error messages are displayed	PASS
TC007	Set auction end time in the past	1. Navigate to Create Auction 2. Set end time earlier	Error message is displayed	PASS

		than current time 3. Submit form		
TC008	Create auction without image	1. Navigate to Create Auction 2. Attempt to upload an image. 3. Submit form	Error message is displayed	PASS

Bidding Tests

Test Case	Description	Steps	Expected Result	Status
TC009	Place valid bid	1. Open auction monitoring 2. Enter bid higher than current highest 3. Submit bid	Bid is accepted and highest bid is updated	PASS
TC010	Place bid below current highest	1. Open auction monitoring 2. Enter bid lower than current highest 3. Submit bid	Error message is displayed	PASS
TC011	Place bid with non-numeric value	1. Open auction monitoring 2. Enter non-numeric value in bid field 3. Submit bid	Error message is displayed	PASS

Payment Tests

Test Case	Description	Steps	Expected Result	Status
TC012	View pending payments	1. Navigate to Make Payment 2. View payment table	Pending payments are displayed	PASS
TC013	Complete payment	1. Navigate to Make Payment 2. Select payment 3. Click Make Payment button	Payment status changes to COMPLETED	PASS
TC014	Automatic payment creation	1. Wait for the auction to end	Payment is automatically created	PASS

		2. Check payment records		

GUI Tests

Test Case	Description	Steps	Expected Result	Status
TC015	Navigation between screens	1. Click navigation menu items 2. Navigate back to main menu	UI transitions correctly between screens	PASS
TC016	Real-time update of auction data	1. Open auction monitoring 2. Place bid from another session 3. Observe UI	Bid information updates automatically	PASS
TC017	Error message display	1. Trigger validation errors 2. Observe error display 3. Fix errors and resubmit	Error messages are displayed and cleared correctly	PASS