Phase II: Design Report for E-Bidding System

Darien Ramdass, Daniel Chen, Evan Haque, MD Rahman

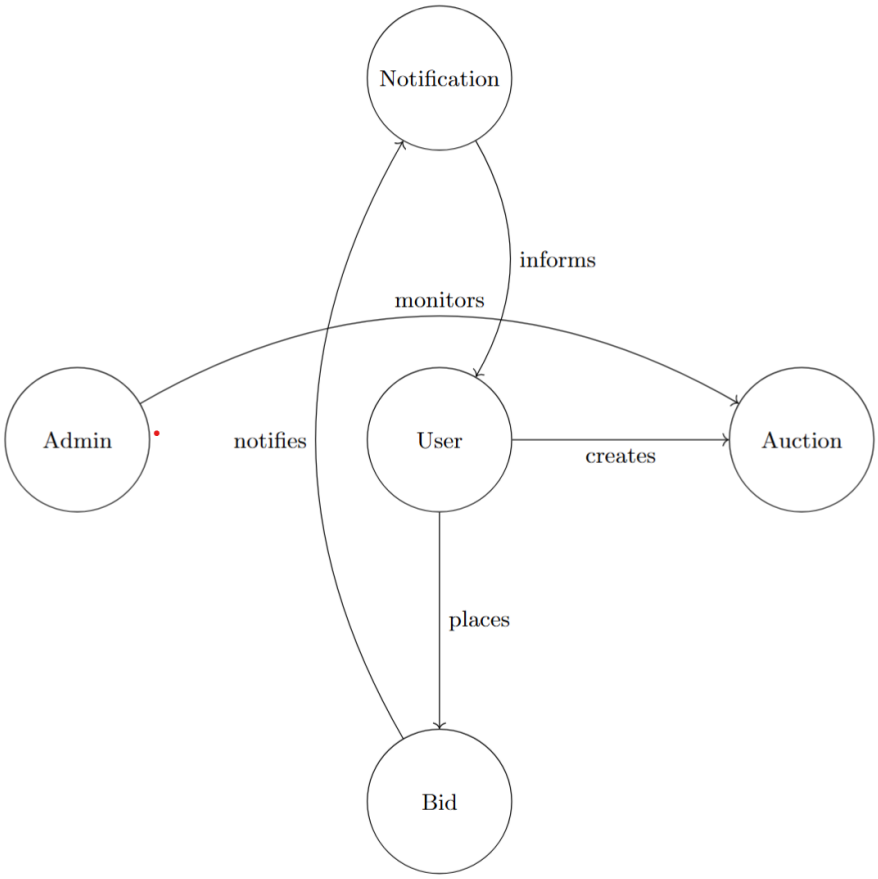
November 12th, 2024

Contents

1 Introduction

This document provides the data structure and logic required to implement the e-bidding system, as dictated by the specification. The following sections outline the collaboration class diagram for an overall system view, use cases with detailed scenarios, and the entity-relationship diagram with attributes and keys for each class. Additionally, the report covers a detailed design using pseudo-code for each method, major GUI screens, and other essential project management aspects.

1.1 Collaboration Class Diagram



The above diagram presents an overall view of how major components of the system interact. The system includes classes such as ‘User‘, ‘Auction‘, ‘Bid‘, ‘Notification‘, and ‘Admin‘, with interactions facilitating functions such as registration, bidding, auction management, and notifications.

2 All Use Cases

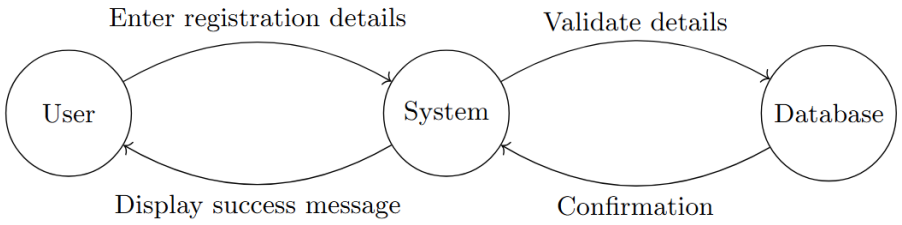
This section presents the scenarios and diagrams for each primary use case of the e-bidding system. Each use case is discussed in terms of normal and exceptional scenarios, along with collaboration or sequence diagrams, and Petri-nets for three selected cases.

2.1 Use Case 1: Register User

Description: This use case allows a new user to register with the system.
Actors: User, System
Preconditions: - The user is not already registered in the system.
Main Flow:
1. The user submits their information, including name, email, and password.
2. The system validates the input data.
3. If the data is valid, the system registers the user and stores the information in the database.
4. The user is shown a success message.

Alternate Flow: - If any information is invalid, the system displays an error message, and the user is prompted to re-enter information.

Since registration is a straightforward process, a collaboration diagram is provided for this use case.

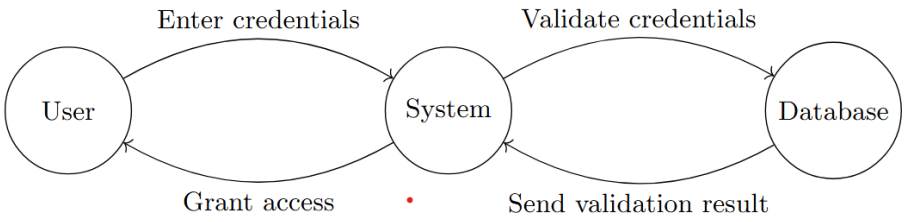


2.2 Use Case 2: Login User

Description: This use case allows a registered user to log into the system.
Actors: User, System
Preconditions: - The user is already registered.
Main Flow:
1. The user enters their credentials (email and password).
2. The system verifies the credentials.
3. If the credentials are correct, the system grants access to the user.

Alternate Flow: - If credentials are incorrect, an error message is displayed, and the user is prompted to try again.

This use case can be represented with a sequence diagram.

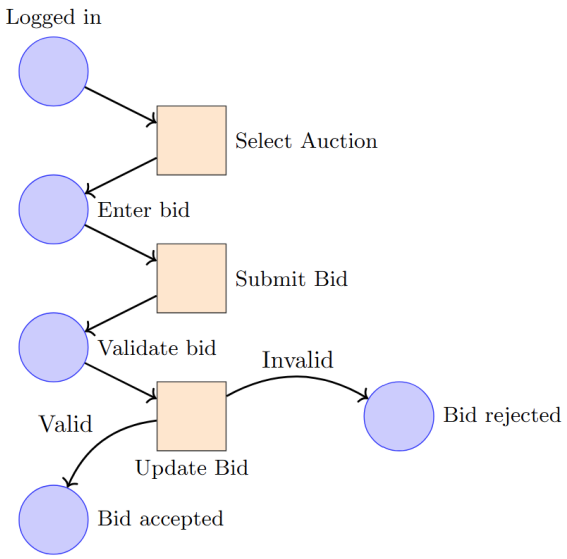


2.3 Use Case 3: Place a Bid

Description: Allows a registered user to place a bid on an auction item.
Actors: Registered User, System
Preconditions: - The user is logged in and the auction is active.
Main Flow:
1. The user selects an auction item and submits a bid.
2. The system verifies the bid amount.
3. If valid, the system updates the highest bid and notifies interested users.

Alternate Flow: - If the bid amount is invalid, the system shows an error message.

A Petri Net diagram is used to represent this process.

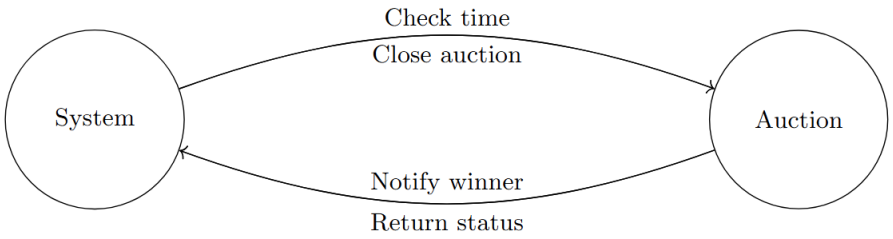


2.4 Use Case 4: Close Auction

Description: This use case allows the system to close an auction once its time has ended.
Actors: System

Preconditions: - The auction's end time has been reached.
Main Flow:
1. The system checks if the auction time has expired.
2. If expired, the auction is closed, and the winner is notified.

This use case is straightforward, so a sequence diagram is appropriate.

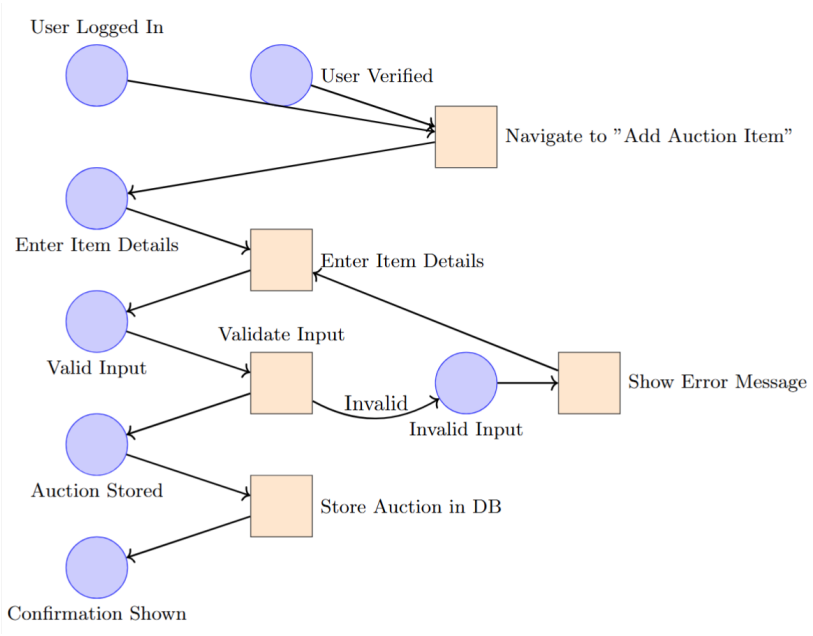


2.5 Use Case 5: Add Item to Auction

Description: Allows a registered user to list an item for auction by providing necessary details such as title, description, starting bid, and reserve price.
Actors: Registered User, System
Preconditions: - The user is logged in. - The user has verified their account (if applicable).
Main Flow:
1. The user navigates to the "Add Auction Item" page.
2. The user enters the item details including title, description, condition, starting bid, reserve price, and auction duration.
3. The system validates the input data.
4. If valid, the system stores the auction item in the database and displays a confirmation message.

Alternate Flow: - If any information is invalid (e.g., missing title or starting bid below a minimum value), the system displays an error message and prompts the user to correct the information.

A Petri Net diagram is used for this use case.

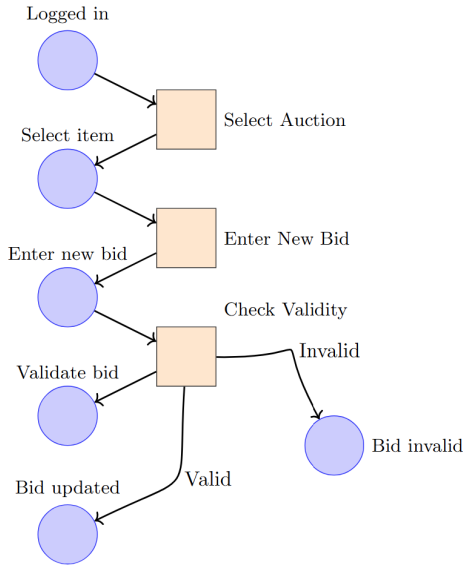


2.6 Use Case 6: Update Bid Amount

Description: This use case allows a user to update their bid amount if they wish to increase it for a particular auction item.
Actors: Registered User, System
Preconditions: - The user is logged in. - The user has already placed an initial bid on the item.
Main Flow:
1. The user selects an item they previously bid on.
2. The user enters a new, higher bid amount.
3. The system validates the new bid amount.
4. If valid, the system updates the bid in the database and notifies other bidders.

Alternate Flow: - If the new bid amount is invalid (e.g., lower than the current highest bid), an error message is displayed.

A Petri Net diagram is used for this use case.

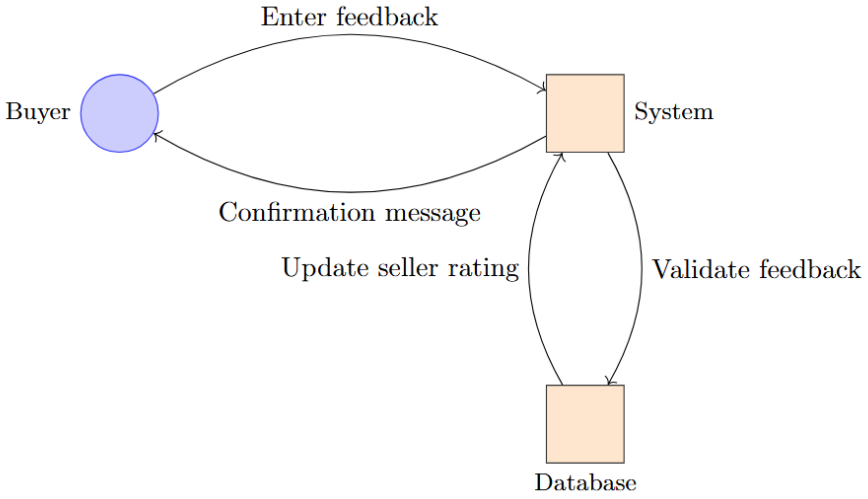


2.7 Use Case 7: Give Feedback on Seller

Description: Allows a buyer to leave feedback for the seller after an auction is completed.
Actors: Buyer, System
Preconditions: - The auction is complete, and the item has been delivered.
Main Flow:
1. The buyer navigates to the completed auction and selects "Leave Feedback."
2. The buyer enters a rating and comment.
3. The system validates the feedback input.
4. If valid, the feedback is stored, and the seller's rating is updated.

Alternate Flow: - If the feedback is invalid (e.g., rating out of range), an error message is displayed.

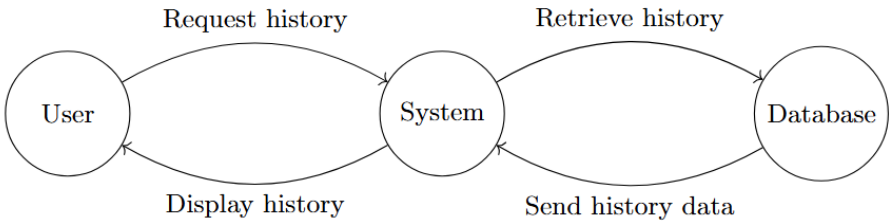
A collaboration diagram is used to represent this use case.



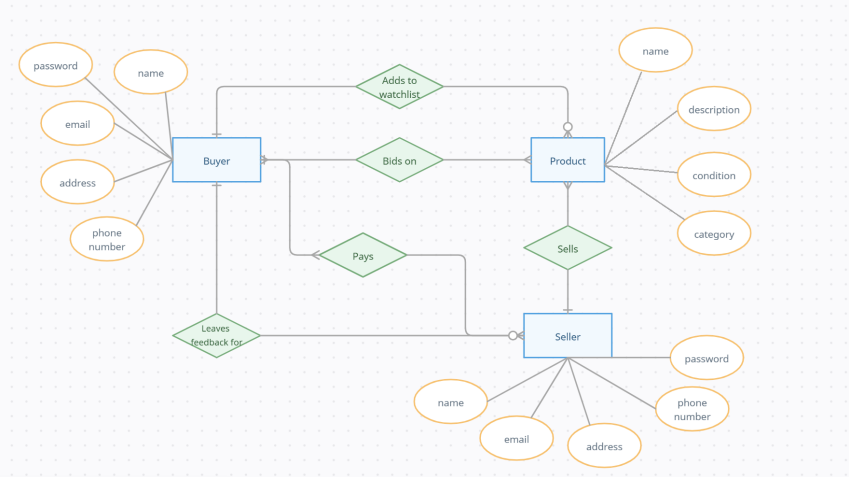
2.8 Use Case 8: View Auction History

Description: This use case allows a user to view their bidding history on past auctions.
Actors: User, System
Preconditions: - The user is logged in.
Main Flow:
1. The user navigates to the "Auction History" section.
2. The system retrieves the user's bidding history from the database.
3. The history is displayed, showing details of each past auction.

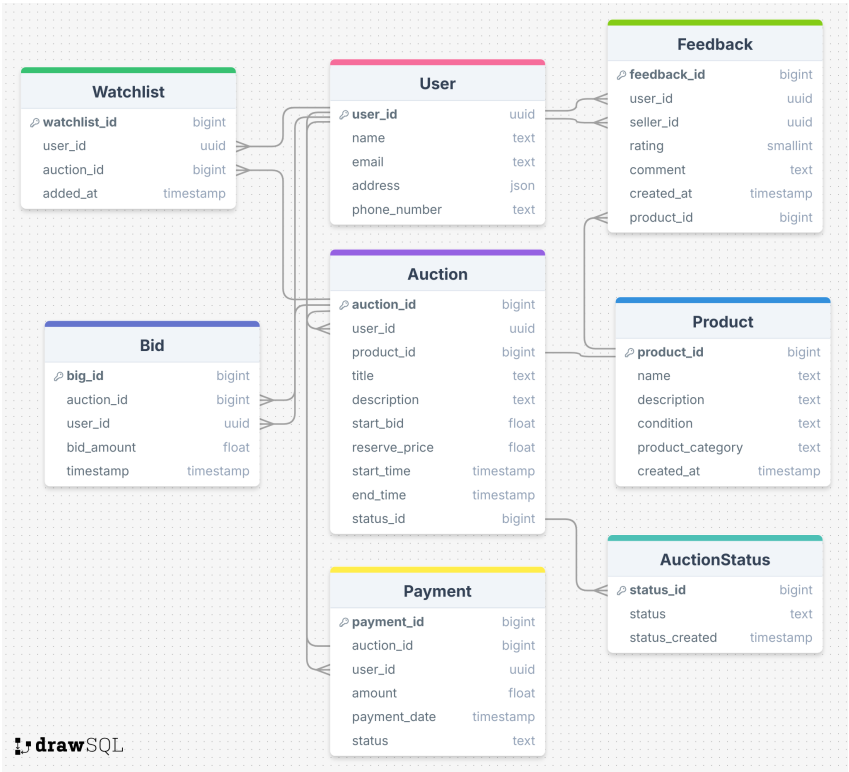
This use case is represented by a sequence diagram.



3 E-R Diagram for the Entire System



ER Diagram



Database Schema

4 Detailed Design

The following pseudocode outlines the input, output, and main functionality for each method in the e-bidding system.

4.1 Register User

Input: User details (name, email, password)
Output: Confirmation of account creation or error message

```
function registerUser(name, email, password):
    if email exists in database:
        return "Error: Email already registered"
    else:
        create new user with name, email, password
        return "Registration Successful"
```

4.2 Login User

Input: Email, password
Output: Authentication token or error message

```
function loginUser(email, password):
    user = findUserByEmail(email)
    if user and user.password == password:
        return generateAuthenticationToken(user)
    else:
        return "Error: Invalid credentials"
```

4.3 Create Auction

Input: Auction details (item, starting bid, end time, user ID)
Output: Confirmation or error message

```
function createAuction(user_id, item, starting_bid, end_time):
    if isValidUser(user_id):
        create new auction with item, starting_bid, end_time, and user_id
        return "Auction Created Successfully"
    else:
        return "Error: Invalid user"
```

4.4 View Auction

Input: Auction ID
Output: Auction details

```
function viewAuction(auction_id):
    auction = findAuctionById(auction_id)
    if auction:
        return auction details (item, highest_bid, end_time)
    else:
        return "Error: Auction not found"
```

4.5 Place Bid

Input: Bid amount, auction ID, user ID
Output: Confirmation or error message

```
function placeBid(user_id, auction_id, bid_amount):
    current_highest_bid = get highest bid for auction_id
    if bid_amount > current_highest_bid:
        record new bid
        update auction's highest bid
        return "Bid Successful"
    else:
        return "Error: Bid amount too low"
```

4.6 End Auction

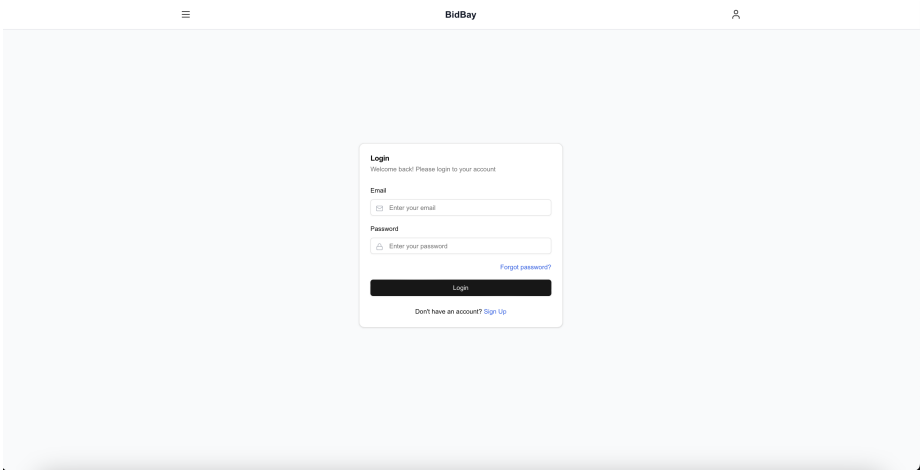
Input: Auction ID
Output: Confirmation or error message

```
function endAuction(auction_id):
    auction = findAuctionById(auction_id)
    if auction.end_time <= current_time:
        declare winner based on highest bid
        close auction
        return "Auction Ended Successfully"
    else:
        return "Error: Auction still active"
```

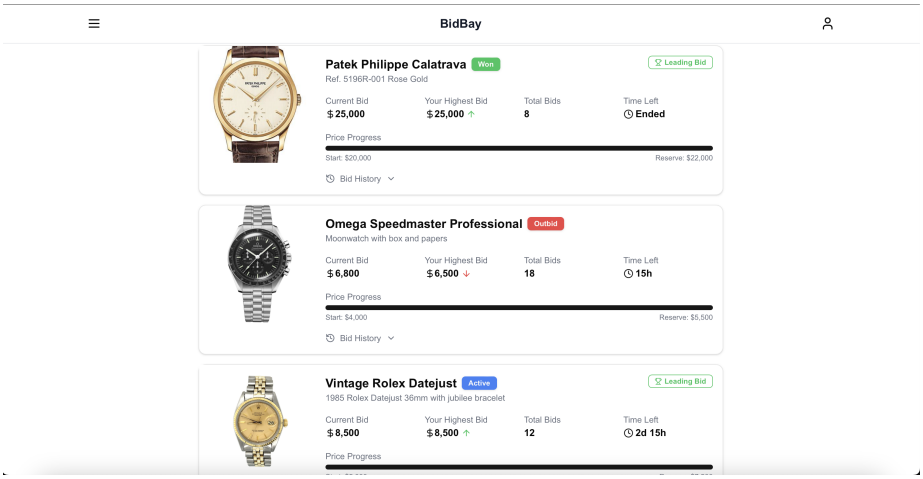
5 System Screens

The system's major GUI screens include:

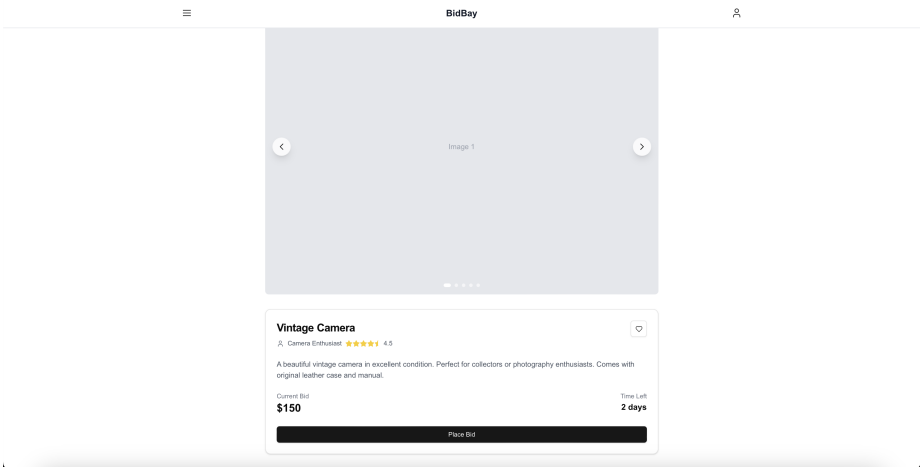
- **Login and Registration Screen:**



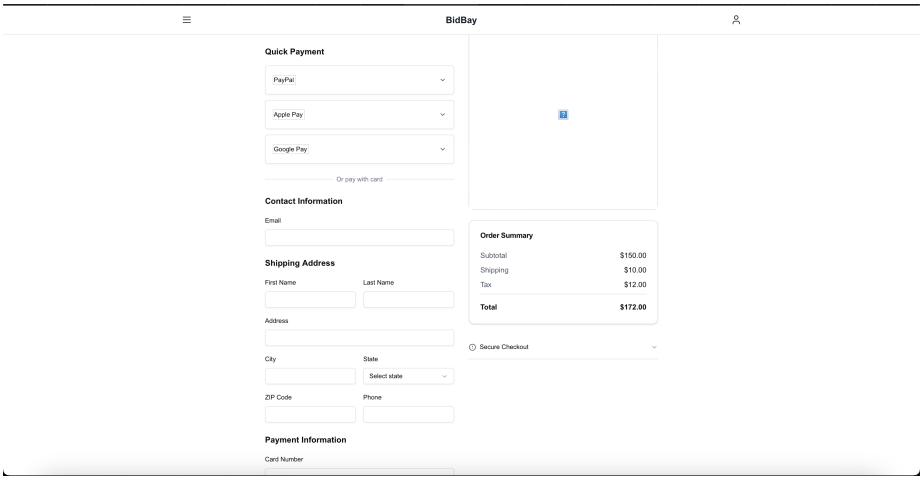
- **Auction Listing Screen:**



- **Auction Details Screen:**



- **Payment Screen:**



6 Memos of Group Meetings and Concerns

- Meeting 1 (10/7): Discussed initial project scope and timeline. Set goals for the next few weeks and assigned initial tasks.
- Meeting 2 (10/10): Reviewed progress on tasks and discussed potential risks. Adjusted project timelines and assigned additional resources to critical areas.

- Meeting 3 (10/14): Discussed system requirements and initial design plans. Assigned roles for system components.
- Meeting 4 (10/17): Reviewed use case diagrams and ER diagrams. Addressed database schema and design concerns.
- Meeting 5 (10/21): Discussed integration points between different system modules. Evaluated security measures and set up testing protocols.
- Meeting 6 (11/5): Finalized implementation details for database and back-end services. Assigned tasks for front-end development.
- Meeting 7 (11/10): Reviewed progress on front-end development. Discussed deployment strategies and set deadlines for final testing.

7 Git Repository

The Git repository for this project is hosted on GitHub and can be accessed at: `https://github.com/dchen024/csc322-project`