Requirements Analysis Document

Michael Buchanon, Bianka Iliev, Jack Daoud, James Dennig

Realtime Commercial Bidding System
Team1

April 25, 2002
## Contents

1 Introduction ............................................. 4  
  1.1 Problem Description .................................. 4  
  1.2 Motivation ............................................ 4  

2 Overview ................................................ 4  

3 Requirements ............................................ 5  
  3.1 Sign up with auction site ................................ 5  
  3.2 Join auction ............................................ 5  
  3.3 Withdraw from auction .................................... 5  
  3.4 Notification ............................................. 5  
      3.4.1 End of auction ...................................... 5  
      3.4.2 New best bid ....................................... 5  
  3.5 Bid placement .......................................... 6  
  3.6 Reserve/Non-Reserve ................................... 6  
  3.7 Bid history .............................................. 6  
  3.8 Auctions start automatically ......................... 6  
  3.9 Create auction ......................................... 6  
  3.10 Auctioneer ............................................. 7  
  3.11 Auctions with no winner ............................... 7  
  3.12 Time and Date formats .............................. 7  
  3.13 Terms of Service ..................................... 7  
  3.14 Auction control ....................................... 7  

4 UML Analysis ............................................. 8  
  4.1 Use Cases .............................................. 8  
  4.2 Object Model .......................................... 13  
      4.2.1 Data Dictionary ..................................... 13  
      4.2.2 Class Diagram ..................................... 16  
  4.3 Dynamic Model ....................................... 17  
      4.3.1 State Diagrams ..................................... 17  
      4.3.2 Sequence Diagrams ................................ 19  

5 Promela Model Verification ................................ 24  
  5.1 Deadlocks ............................................. 24  
      5.1.1 newbid equals maxbid problem ..................... 24  
      5.1.2 client recieves end auction problem .............. 25  
  5.2 Linear Temporal Logic ................................ 26  

6 Appendix A .............................................. 28
**List of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UML Use Case Diagram</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>UML Class Diagram</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Extended State Machine for Auction Site</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Extended State Machine for Auctioneer</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Sequence Diagram for Failed Login</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Sequence Diagram for Joining an Auction</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>Sequence Diagram for Creating an Auction</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>Trace from XSpin</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>Trace from XSpin</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>LTL Trace from XSpin</td>
<td>27</td>
</tr>
</tbody>
</table>
1 Introduction

With the advent of widespread Internet use among the general public, many traditional commercial
companies have found a new medium to make their products and services more widely available.
One of the more popular services to come about is online auctions.

The following document is a requirements analysis for a Realtime Commercial Bidding System.
The purpose of this document is to reveal the functionality, components, and rules by which our
Realtime Commercial Bidding System will be built. The requirements analysis will be divided
into four distinct sections: introduction to the domain, overview, specific requirements, and UML
analysis of the components and user interactions within the domain of the Realtime Commercial
Bidding System. The document includes use-case diagrams, an object model consisting of a class
diagram, dynamic models, and state diagrams to thoroughly illustrate system functionality, and
sequence diagrams showing the flow of events for particular scenarios.

1.1 Problem Description

We will be designing the user interface and system functionality for our distributed online brokering
system. The system will consist of five main components. The needed components are the Auction
Site, Auction, Auctioneer, User, and Bids. The auctioneer is the automaton that runs the auction
and is the primary facilitator between the creator of the auction and the bidders involved in the
auction. The auction creator is the individual that initiates the brokering of a good or service. It
is they who determine the characteristics of the auction. The bidder is the entity that makes an
offer on the good or the service.

1.2 Motivation

Many distributed online brokering systems have already found great success and established that
there is a market for this type of service on the Internet. However, none are run in real time or
offer the user enough up to date information as to simulate a real time auction. Our Realtime
Commercial Bidding System will offer the auction creator and the bidders current information on
the state of the Auction in real time. Our system also offers the auction creator a greater variety
of options as to how the auction will be conducted. Specifically, our system allows the auction
creator to not only sell a good or service, but also to solicit bids for a good or service that they
are interested in obtaining. This concept is referred to as a reverse auction. With all of the sales
taking place, it is easy to see where a fair amount of revenue could be garnered from charging even
a modest fee for this service.

2 Overview

The Realtime Commercial Bidding System is structured around the concept of an auctioneer. The
auctioneer acts as the referee between the auction creator and the bidders for the auction site,
lowering the need for human staffing. When an auction is created the auctioneer conducts the
auction based on the characteristics defined by the auction creator. When the bidder makes a bid
the auctioneer receives the bid and determines if the bid is acceptable. If the bid is acceptable and
better than the previous best bid, the auctioneer informs the auction creator and the bidders that
there is a new best bid and what that bid is. The bidder is also informed when they currently have
the best bid. If the bid is unacceptable, only the bidder is informed and the state of the auction
is not changed. The auctioneer mediates these exchanges throughout the duration of the auction.
At the end of the auction, all parties are notified of the end of the auction and told who won. The auctioneer will also contact the winning bidder and auction creator, via their preferred method of contact, with information about how to finalize the transaction.

3 Requirements

3.1 Sign up with auction site

The user shall provide contact information such as a phone number, address, and payment information. The user will also provide a username and password, which will be used to login to the auction site. All of this information shall be stored on a central server maintained by the auction site administrators. This information will be used by the auction site for validating logins, contacting users, and billing users.

3.2 Join auction

In order to bid, a user of the system must join an auction. From conversations with the customer, this can be before or after an auction has started. A list of current and future auctions will be provided after the user logs into the auction site. The user can select any auction and choose to join that auction. If a user creates an auction, they can join it to monitor the status. However, they will not be allowed to bid.

3.3 Withdraw from auction

A user in an auction can choose to withdraw. If the user's bid is currently the best bid, they will be notified that their bid will still be valid after their withdrawal. Otherwise, the bidder will be allowed to withdraw from the auction at will.

3.4 Notification

3.4.1 End of auction

The auctioneer keeps a record of all the users and the two best bidders in order to determine the winner of the auction. If the primary best bidder is disqualified, the second best bidder will then be the winner. If someone wins the auction, then all the active bidders including the winner and the auction creator will be notified. The auction site will be notified of the end of the auction and the winning price. Notification will display on the bidder's screen to inform them that the auction has ended and who has won. If the auction ends with no winner, all parties will be notified of this as well.

3.4.2 New best bid

From the customer specification, we shall keep track of the highest bid in a straight auction and the lowest bid in a reverse auction. This is so that when the auction ends, either by timeout or by reaching the designated end time, we can determine a winner. We shall also keep track of the second best bid in any given auction so that the auctioneer can declare that bidder the winner in the event that the current best bidder is disqualified. From conversations with the customer, we know that we should notify all bidders when the best bid has changed. This shall be done via a
notification window on the bidder's screen. The bidder's identity shall remain anonymous to all but the creator of the auction until the end of the auction if a winner is declared.

3.5 Bid placement

All registered users of the auction system may have access to their current list of bids and must be able to join an auction. Bids are received by the auctioneer from bidders in real time. The auction creator shall not be able to bid in the auction that they have created. All currently active bidders will be notified about the best bids every time there is a new best bid. If the bid is not the best bid, it is ignored by the auctioneer. In either case, the bidders' bid history is updated to reflect the submitted bid. If two or more bids are received by the auctioneer at the same time, known as concurrent bidding, on of these bids will be picked at random to avoid lost bids.

3.6 Reserve/Non-Reserve

The auction can be in two different modes, "Reserve" and "Non-Reserve" mode. In a reserved auction, a minimum or maximum price must be met to complete a sale, depending on the type of auction. The seller will not sell their item if this reserve price is not met. If the auction is a "Non-reserve" auction, then the best bidder wins regardless of the bid amount when the auction ends.

3.7 Bid history

From talking with our customer, it has become clear that the bidder's interface to the auction site will keep a list of bids for the auction in which they are currently active. The customer would also like to keep a running history for each user of the system. This history will consist of all the bids and auctions participated in. These past bid histories are to be stored on a server along with the rest of the bidder's contact information. They will be retrieved when the participant logs into the auction site.

3.8 Auctions start automatically

From the specification, auctions are allowed to be scheduled to start in the future. When an auction creator creates an auction, they specify the time and date that the auction will start. The defaults are the current time and the current date. In the default case, once created the auction would be active. If the creator specifies a date or time in the future, the auction would start at that time. Bidders for the created auction will not be able to place bids until after the auction starts. If a bidder attempts to place a bid before the auction has started they will receive a notification that the auction has not yet started. When the time has arrived for the auction to start all bidders currently waiting and the creator will be notified of the start of the auction.

3.9 Create auction

From the specifications requested by the customer, the auction site users must have the option of creating an auction so that they may sell items or request goods and or services. The auction site should allow two types of auctions; the traditional auction in which a client sells an item to a group of bidders for the highest possible price, or a reverse auction in which a bidder will be buying a certain items or services from a group of other bidders. In both cases the automated auctioneer will mediate these auctions. As shown in the sequence diagrams, after the user logs
in and is authorized they should have two choices; join a current auction or create an auction of their own. If they decide to create an auction, then they should supply a name, description, start and end times, start and end dates, initial price(if any), the reverse price(if any) and if it is a normal or reverse auction. The customer has asked that the users of the auction site be able to configure the amount of ”dead-time” before the auction ends. By default, this ”dead-time” will be set to ten minutes. If this ”dead-time” elapses and there is no activity in the auction, it will end automatically. Otherwise, the auction will end at the specified by the auction creator.

3.10 Auctioneer

We shall provide an automated auctioneer in order to reduce staffing requirements and to make this system as simple to run as possible. The auctioneer will keep the bidders and creators notified of all that happens in the auction by giving them status updates on their screens. From the class model, it is clear that there is one auctioneer for every auction. The auctioneer will also implement any business logic that is needed for the auction to proceed, such as labeling the lowest bid the best bid in a reverse auction or the highest bid the best bid in a normal auction.

3.11 Auctions with no winner

From our conversations with the customer, it has been determined that there is the possibility of an auction ending with no winner. This can happen if the reserve is not met or if the auction ends with no bids being made, either through reaching the end time or through inactivity for longer than the length of the ”dead-time”. Then all bidders and the creator will be notified that there is no winner.

3.12 Time and Date formats

From the prototype we have found that a time and date format must be specified for reasons of consistency. Times will be of the form: HH:MM AM or HH:MM PM and dates will be of the form: MM/DD/YYYY

3.13 Terms of Service

To satisfy legal requirements, we will have a Terms of Service that users of the system must agree to during registration before they can use the system. If the Terms of Service changes, the user of the system will be required to agree to them again at login.

3.14 Auction control

There must be a way for a human administrator to control an auction. This will allow the administrators to remove auctions as well as users that are deemed inappropriate.
4 UML Analysis

4.1 Use Cases

The purpose of the Use-Case model in Figure 1 is to demonstrate how the whole Bidding system interacts with the actors and sub-systems that are a part of it. The Use-Case diagram consists of a system boundary that is represented by the rectangle in the figure. The idea behind the system boundary is that it separates the actors and services in the outside environment from the services provided by the system. Actors are the entities that interact with the system. In the bidding system we have the User, Administrator and Security as actors. As shown in the figure 1, the lines between the actors and the use-cases shows the association between them. Inside the bounded system we see lines connecting the different use-cases. This describes how these use cases are related to each other.

Figure 1: UML Use Case Diagram
<table>
<thead>
<tr>
<th>Use case</th>
<th>bid on item</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>The user places a bid on the item up for auction. The Auctioneer notifies the user when he/she becomes the highest bidder and if they have won the auction when it is over.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Bid placement</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>authorization</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>get list of auctions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>After the User logs in, a list of auctions is available for the user to join.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Join auction</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>get bid history</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>After the User joins an auction, a list of bids the user has made is available to them.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Bid history</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
<tr>
<td>Use case</td>
<td>join auction</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>After the User logs in, the user chooses an auction from the list available and joins.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Join auction</td>
</tr>
<tr>
<td>Includes</td>
<td>none</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Security</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>Is part of the login process. Checks to see if username and password are valid.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Sign up with auction site</td>
</tr>
<tr>
<td>Includes</td>
<td>none</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>User, Administrator</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>User enters username and password, which is verified by Security.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Sign up with auction site</td>
</tr>
<tr>
<td>Includes</td>
<td>Authorization</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
<tr>
<td>Use case:</td>
<td>withdraw</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>User decides to withdraw from an auction.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Withdraw from auction</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case:</th>
<th>remove user from auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>Administrator</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>A User will be removed from an auction whenever he wishes. If his bid is the highest he will be notified that it will remain valid after his withdrawal.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Withdraw from auction</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case:</th>
<th>cancel auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>Administrator</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>An auction is deleted from the list of auctions.</td>
</tr>
<tr>
<td><strong>Cross-reference:</strong></td>
<td>Auction control</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
<tr>
<td>Use case</td>
<td>create account</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>User creates a new account with the auction site which will be used for all future participation.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Sign up with auction site</td>
</tr>
<tr>
<td>Includes</td>
<td>Authorization</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>create auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>User creates an auction to sell an item.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Create auction</td>
</tr>
<tr>
<td>Includes</td>
<td>none</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>monitor auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Administrator</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description</td>
<td>A User that creates an auction can monitor the events of that auction. The auctioneer will keep the User up to date regarding the status of the auction.</td>
</tr>
<tr>
<td>Cross-reference</td>
<td>Join auction</td>
</tr>
<tr>
<td>Includes</td>
<td>none</td>
</tr>
<tr>
<td>Extended by</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>
4.2 Object Model

4.2.1 Data Dictionary

Auction Site: A collection of auctions past, present and future.

- Properties
  1. auctionList: A list maintained by the Auction Site of the currently available auctions.
  2. knownUserList: List of users that are currently logged into the system.

- Methods
  1. CreateAuction(): A method in the Auction Site used to spawn an Auctioneer for a given auction based on information supplied by the user.
  2. GetAuctionHistory(User): A method in the Auction Site that returns the auction history for the given user.
  3. ListAuctions(): A method in the Auction Site that enumerates the current auctions.
  4. RegisterUser(User): Allow a first time user to register with the auction site.
  5. SaveAucInfo(User, Bid, Time): This method is used to record into the database who won, what their bid was and at what time the auction was won.

Auction: An event run by the auctioneer to sell or buy goods and/or services.

- Properties
  1. auctionID: A unique string by which a specific auction is identified by the auction site.
  2. endTime: When the auction ends.
  3. initialPrice: The price at which the auction will start.
  4. isRegular: Boolean variable that is true if the auction is a normal auction and false if it is a reverse auction.
  5. itemDesc: A description of the item up for auction.
  6. reserve: The minimum price that a normal auction will sell for or the maximum price a reverse auction will sell for.
  7. startDate: The date on which the auction starts.
  8. startTime: Time when the auction starts.
  9. timeOut: If set, the Auctioneer will end the Auction after this many minutes of inactivity. The default is 10 minutes.

- Methods
  1. isReserve(): A method of the Auction that returns true if the ReservePrice is greater than $0.00 and false otherwise.
**Auctioneer** : Automation that controls the auctions, notifies the users of the new best bid, and notifies the winner of the auction if one exists.

- Properties
  1. auctionCreator: A reference to the user object that created the current auction.
  2. bestBid: Current Bid that is the highest in a normal auction and the lowest in a reverse auction.
  3. bestBidUser: current user object that has the best bid.
  4. listOfUsers: A list of all users currently in the auction.
  5. nextBestBid: The second best bid in the current auction tracked in case some circumstance invalidates the current bestBidUser.
  6. nextBestBidUser: the user object that has the second best bid.

- Methods
  1. Bid(): A function in the auctioneer that accepts a bid from a user.
  2. JoinAuction(user): Adds the user to the current auction.

**Bid** : Object representing a placed bid in the auction system.

- Properties
  1. amount: The amount of the bid.
  2. timeStamper: A fine grained time stamp used by the system to determine when the bid was placed.
  3. userID: The ID of the user who made the bid.

- Methods
  1. None

**User** : An object representing the various people who will use the system.

- Properties
  1. address: A string holding the address of the user.
  2. bidHistory: A history of all the bids that a particular user has placed.
  3. creditSts: A representation of the users credit status. It can be good, questionable, bad, unknown.
  4. emailAddress: The users email address.
  5. fullName: A string holding the users full name.
  6. userName: A unique identifier that is used to track all customers of the Auction Site.
• Methods
  
1. dispAucWnd(auctionID, bool): A method used to display the current auction to the end user. If the bool is true, then the user is the auction creator and they will just be able to view the auction.

2. dispAucWnd(auctionList): A method used to allow the user to select from a list of current auctions. The auctionList is the list of auctions retrieved from the auction site.

3. dispTOS(): This function displays the Terms Of Service for the auction site. The language of this TOS will be supplied by the lawyers of the company running the site.

4. isAdmin(): Checks with the Security module to determine if the user is a site administrator and returns true if they are and false otherwise.

5. remove(userName): Allow an administrator to remove a user from an auction.

6. remove(auctionID): Allow an administrator to remove or cancel an auction registered with the auction site.

Security: Security program installed to protect users privacy and auction site. Consists of encryption/decryption schemes and functionality to validate bids and users.

• Properties
  
1. none

• Methods

1. Decrypt(Object): Generic method to decrypt information after it's been encrypted.

2. Encrypt(Object): Generic method to encrypt data and objects.

3. Validate(Bid): returns true if the bid is coming from a valid client and false otherwise.

4. Validate(User): returns a true if the user has logged in and passed the security checks and false otherwise.
4.2.2 Class Diagram

The Class Diagram in Figure 2 shows all the classes in the Real Time Bidding system including all the attributes and operations present within each class. The diagram also shows the associations between the different classes which describes how the classes relate with each other. The class diagram consists of six classes: Auction, Auction Site, Auctioneer, Bid, Security, and User.

The User class contains the attributes and operations related to the user. After a user registers with the auction site and becomes a valid user they can make a bid as can be seen in the diagram through the association with Bid class. The user can create an auction and be constantly updated through the Auctioneer and can also view the auction list and known user list through the auction site. The Bid class is an aggregate of the Auctioneer class and and describes the bid in terms of the amount, time of the bid and the user making the bid.

The Auctioneer class describes the attributes and operations of the Auctioneer which monitors the auctions and notifies the users through the Auction site of any updates. The Auction Site class interacts with the User, Auction and Auctioneer classes and Auction class describes a certain auction and is an aggregate of the Auction Site class.

Finally, the Security class plays a major role since the user can not proceed before passing this class.

![Figure 2: UML Class Diagram](image-url)
4.3 Dynamic Model

4.3.1 State Diagrams

Figure number 3 represents the state diagram of our Realtime Bidding System. It consists of states and transitions labeled with the events, guards, and actions that occur when the transition is taken. The states are represented in the diagram by rectangles with curved corners while the transitions are the lines that can be seen between the states. As can be seen on the diagram some brackets exist with certain conditions in them, these are known as guards. A guarded transition will only be taken if the conditions evaluate to true. An absent guard is the same as a true guard.

The starting point of the state diagram is the **Idle** state. The user has to register with the site before he can participate in any auction activity. Once the user submits his login information, that takes us to the next state which is the **Verification** state. If the user is an invalid user he is sent back to the **Idle** state otherwise he is transferred to the **Logged in** state. When the user is in the **Logged in** state, a list of auctions is presented and the user can choose to join an auction, create a new auction, or quit. If the user quits they go back to the **Idle** state. If They decide to join an auction then they are transferred to the **Join Auction** state, if they decide to quit after bidding, this takes them back to the **Logged in State** if they are not the best bidder. If the user is the best bidder, then they transition to the **Warning** state will occur warning the user that their bid will still be valid even after quitting.

The user can create a new auction, if the created auction is not a valid auction they will be notified. If it is a valid auction then a transition to the **Join Auction** state will take place. If the user quits then a transition back to the **Logged in** state occurs. From the **Logged in** state they can quit and exit the auction site or login back again.

![Figure 3: Extended State Machine for Auction Site](image-url)
In Figure 4 we show the dynamic model for the auctioneer. The auctioneer starts in the Idle state. There are only two ways the auctioneer moves out of the Idle state. The first is when the auctioneer receives a bid from a user. When a bid is received the auctioneer moves to the Check bid state. This is where most comparisons are made within the auctioneer to determine the next course of action.

The first case is where the bid is not a valid bid. An example of this case would be that the format of the bid is incorrect, such as a user entering a series of letters instead of a dollar amount. In this case the user is notified that it is an invalid bid and the auctioneer returns to the Idle state.

The next case is when a valid bid is received from a user that is the auction creator. The auction creator is not allowed to bid in their own auction. In this instance, the user is notified that their bid will not be accepted because bids from the creator are not allowed.

Another case is when a valid bid is received but the bid does not beat the current best bid. This occurs when the auction is a standard auction and the bid received is lower than the current best bid. It also happens when the auction is a reverse auction and the bid is higher than the current best bid. When this occurs, the user is notified that their bid is not the best bid and the auctioneer returns to the Idle state.

The last scenario that can occur from the Check bid state is when a valid bid is received and it is better than the current best bid. This occurs in a standard auction when the bid is higher than the current best bid, and it occurs in a reverse auction when the bid is lower than the current best bid. When this occurs the user is notified that they now have the best bid and the rest of the users are notified that there is a new best bid. The auctioneer then returns to the Idle state.

The second way that the auctioneer moves out of the Idle state is when the dead time event has occurred. The dead time event occurs when there has been no activity within the auction and the auctioneer has been in the idle state for the amount of dead time set by the auction creator. When this happens the users are notified that the auction has ended and whom the winner is if there one.

![Extended State Machine for Auctioneer](image-url)

**Figure 4: Extended State Machine for Auctioneer**
4.3.2 Sequence Diagrams

Figure 5 shows what happens when a user attempts to log in to the auction site and fails.

1. The User connects to the website (the Auction Site itself)
2. The Auction Site requests user to login, and requests username and password with the Login() method
3. The User accepts and submits information
4. The Auction Site requests verification of username and password from Security
5. Security signals to the Auction Site that the User is invalid (!validUser)
6. "Invalid Login" message displayed to User

![Sequence Diagram for Failed Login](image-url)

Figure 5: Sequence Diagram for Failed Login
Figure 6 shows what happens when a user joins an auction and bids on an item.

1. The User connects to the website (the Auction Site itself)

2. The Auction Site requests user to login, and requests username and password with the Login() method

3. The User accepts and submits information

4. The Auction Site requests verification of username and password from Security

5. Security signals the Auction Site that the User is valid (validUser)

6. The Auction Site displays a list of auctions

7. User Selects an Auction to join, and joins the auction

8. Auction Site displays bid/auction window to User

9. User Makes Bid and it is sent to Auctioneer

10. Auctioneer validates bid via Security

11. Security signals to Auctioneer that the bid is valid

12. Auctioneer notifies the user of the new best bid

13. User requests to withdraw from the auction

14. Auctioneer notifies the User that they are the current best bidder and that they are still responsible for purchasing the item if they remain the best bid when auction ends
Figure 6: Sequence Diagram for Joining an Auction
Figure 7 shows what happens when a user attempts to Create an Auction.

1. The User connects to the website (the Auction Site itself)
2. The Auction Site requests username and password with the Login() method
3. The User accepts and submits information
4. The Auction Site verifies username and password via Security
5. Security signals to Auction Site that User is valid (validUser)
6. The Auction Site displays the Join/Create window
7. The User selects Create and sends auction information to the Auction Site
8. The Auction Site requests for auction information; displays the Create Auction Window
9. The User submits the Auction information to the Auction Site
10. The Auction Site creates the Auction and the Auctioneer
11. The User makes a bid and is sent to the Auctioneer
12. The Auctioneer validates bid via Security
13. Security notifies Auctioneer that the bid is not valid (because creator of an auction cannot bid in that auction).
14. Auctioneer notifies the user that the Auction Creator cannot bid.
Create an Auction Sequence Diagram

User

- connect to website
  - Login()
  - submit
  - ListAuctions()
  - create
    - dispAucWnd()
  - submit
    - dispAucWnd(auctionID, bool)
  - Bid(amount)

AuctionSite

- Login()
- ListAuctions()
- create
- dispAucWnd()
- submit
- dispAucWnd(auctionID, bool)
- Bid(amount)

Auctioneer

- JoinAuction(User)

Security

- validUser
- Validate(User)
- !validBid
- Validate(Bid)
- !validBid

Figure 7: Sequence Diagram for Creating an Auction
5 Promela Model Verification

5.1 Deadlocks

deadlock: A situation wherein two or more processes are unable to proceed because each is waiting for one of the others to do something.

5.1.1 newbid equals maxbid problem

As you can see from Figure 8, when the auctioneer model reaches the Save Bid state, it sends the ack event to the client right away. This allows the client to move on, however the auctioneer is stuck in the state Save Bid because it does not have a transition to take when newbid equals maxbid. This results in a deadlocked system. We tested this with XSpin using assertions. See Appendix A for PROMELA code.

Data Values from XSpin:

maxbid = 8
newbid = 8
state = saSave_Bid
state2 = scIdle

Figure 8: Trace from XSpin
5.1.2 client receives end auction problem

If the client receives the *end_auction* event while in any state except for the **Idle** state it will cause the client to enter a **deadlock**. It does this because the only place it has instructions on how to handle an *end_auction* event is in the **Idle** state. When this happens the auctioneer will successfully reach the **End_All** state while the client will be stuck in whatever state it was in when it received the *end_auction* event.

As you can see from Figure 9, the trace shows the auctioneer receiving the *auction_time_elapse* event and issuing the *end_auction* event to the client. The client is in the state **User_Bid** so it does not have a valid transition out of the state on the event it has received. The auctioneer moves on to **End_All** state while the client is **deadlocked** in **User_Bid**.

Data Values from XSpin:

```plaintext
maxbid = 8
newbid = 10
state = saEnd_All
state2 = scUser_Bid
```

![Figure 9: Trace from XSpin](image-url)
5.2 Linear Temporal Logic

Demonstrated here is a liveliness property failure. The property we test is: if a user sends a bid request then eventually a bid will always be sent. The formula is set up as:

\[ \square(P \implies \Diamond Q) \]  

(1)

Where P and Q are defined as:

\begin{align*}
P &= \text{event[user\_bid\_request]} \\
Q &= \text{state2 == scSent\_Bid}
\end{align*}

(2) (3)

This formula fails under LTL analysis. As you can see from Figure 10, the trace leads to the client being in state Get\_Bid and that is where it stops. So the state Sent\_Bid was never reached and our LTL formula has failed.

Data from XSPin:

\begin{verbatim}
maxbid = 8
newbid = 4
state = sWait
state2 = scGet\_Bid
\end{verbatim}
6 Appendix A

/* States */
/*client side*/
mtype {scIdle, scHi_Bid, scUser_Bid, scGet_Bid, scSent_Bid, scTerminal};
/*auctioneer side*/
mtype {saWait, saCheck_OK, saSave_Bid, saHig_Bid, saEnd_All, saTerminal};
/*client events*/
mtype {high_bid, end_auction, user_bid_request, bidok, auc_timeout, ack};
/* auctioneer events */
mtype {bidrqst, bidmsg, auction_time_elapse};
/* button events */
mtype state;
mtype state2;

/* channels for transitions */
chan event = [0] of {mtype};

int maxbid = 8;
int newbid = 8;

/* ========= straight up sequence ========= */
init
{
/*do testing stuff here*/

run auctioneer();
run client();

event!user_bid_request;
printf("Sent user bid request");
timeout;
assert(state == saWait);
active proctype auctioneer()
{

state = sWait;
do :: (state == sWait) -> printf("Auctioneer in state Wait\n");
if :: event?bidrqst -> event!bidok -> state = sCheck_OK;
   :: event?auction_time_elapse -> state = saEnd_All;
fi;
   :: (state == sCheck_OK) -> printf("Auctioneer in state Check_OK\n");
if ::event?bidmsg -> state = saSave_Bid;
   :: event?auction_time_elapse -> state = saEnd_All;
fi;
   :: (state == saSave_Bid) -> printf("Auctioneer in state saSave_Bid\n");
   event!ack;
      printf("Sent ack\n");
if :: newbid < maxbid -> state = sWait;
   :: newbid > maxbid -> state = sHig_Bid;
   :: event?auction_time_elapse -> state = saEnd_All;
fi;
   :: (state == sHig_Bid) -> printf("Auctioneer in state High_Bid\n");
      maxbid = newbid;
state = sWait;
event!high_bid;
if :: event?auction_time_elapse -> state = saEnd_All;
fi;

:: (state == saEnd_All) -> printf("Auctioneer in state End_All\n");
   event!end_auction;
state = sTerminal;

:: (state == sTerminal) -> printf("Terminal State Auctioneer\n");
break;


od;
}

active proctype client()
{
    state2 = scIdle;

do :: (state2== scIdle) -> printf("Client in state Idle\n");
    if :: event?high_bid -> state2 = scHi_Bid;
        :: event?user_bid_request -> state2= scUser_Bid;
        :: event?end_auction -> state2 = scTerminal;
    fi;
        :: (state2 == scHi_Bid) -> printf("Client in state Hi_Bid\n");
        printf("Displaying High Bid\n");
    state2 = scIdle;
        :: (state2 == scUser_Bid) -> printf("Client in state User_Bid\n");
    event!bidrqst ;
        if :: event?bidok -> state2 = scGet_Bid;
    fi;
        :: (state2 == scGet_Bid) -> printf("Client in state Get_Bid\n");
        printf("Getting bid amount\n");
    event!bidmsg;
    state2 = scSent_Bid;
        :: (state2 == scSent_Bid) -> printf("Client in State Sent_Bid\n");
    if :: event?auc_timeout -> state2 = scUser_Bid;
        :: event?ack -> state2 = scIdle;
    fi;
        :: (state2 == scTerminal) -> printf("Terminal Client\n");
    break;
    od;
}