POLITECNICO DI MILANO

Course in Computer Science and Engineering
Department of Electronics, Informatics and Bioengineering

GuessBid Project

Requirements and Specification Document

Authors:
Ervin Kamberoski
Pavel Gichevski
Egzon Ademi

Supervisor:
Dr. Elisabetta Di Nitto

April 27, 2015

Contents

List of Figures

1 Introduction

1.1 Problem Description

GuessBid is an application implementing an inverse auction system. An inverse auction works like a regular auction. The difference is that in an inverse auction a user has to propose the lowest unique bid to win the auction. Each user has to guess the lowest bid. Bids can be placed till the bidding closing time. Then they are evaluated by the system. The winner is the user who has placed the lowest unique bid.

1.2 System Features

In this section we will provide a combination of the system features and goals, as we think that there is no line that clearly separates the two.

- *Registering a new user:* This is one of the main goals that the system must realize, to be considered as functional.
- *Creating a new auction:* A user can create a new auction of a good that he/she possesses.
- *Listing auctions:* The system must provide to the user, the ability to list the auctions that are in course.
- *Bids on auction:* This is an obvious feature, by which the user puts their bid for a specific good being offered by some other user.
- *Auction chat:* For each auction, the system will provide a chat session whose participants are all users who has made a bid for it.
- *Auction history:* The page which provides history of the outcomes of the auctions to which the current user is related.
- *Auction notifications:* The system must notify the user of the outcome of the active auctions, either as participant or as owner.
- *Bids status:* Every user will be informed about the actual status for the bids to auctions opened by other users.

1.3 Domain Properties

In this section we list a sequence of properties that the system supposes from the world for its correctness and reliability.

- The personal information that the user provides are correct. We need this assumption for the whole integrity of the system.
- A user creates auction of a good he/she intends to sell. Once the auction is created it cannot be canceled.
- A user creates auction of a good he/she actually possesses.
- A user bids to an auction if they are actually convinced to take the good in case they win the auction.
- We assume fair usage by the user, in the sense that a single user won't create multiple accounts to help themselves win a certain auction.

1.4 Technical Terms

In this section we provide a list of technical words that are known by the system, and will be used throughout the document. Although some of the words may seem obvious from general knowledge, they may have a specific meaning and interpretation in the system domain.

- **Auction.** A procedure through which a user makes available one of his goods, to the other ones. As previously mentioned, the auction is won by the smallest-unique bid.
- **Administrator.** A person who manages the system. Controls its behavior, repairing eventual problems that may arise.
- **User.** A person who has undergone the registration process, and is part of the system.
- **Auction Feed.** A page where active auctions are listed. The user can list a specific type of goods by using adequate filters, or see all of them.
- **Frequently Asked Questions.** A dedicated section of the web application, where some known issues are explained and where most frequent questions are answered. Before asking for any help, the user is advised to access this section.

- **Auction bid.** Action through which the users joins the competition for the offered good, by paying 2 units of credit.
- **Auction chat.** Auction specific area, where users that have bid to the auction can share impressions or exchange opinions regarding the good.
- **History of auctions.** Refers to a place where all actions which are related to the current user somehow, are listed.
- **Bid Status.** It can be either winning or losing depending of the fact whether auction was won by that bid or not.

2 Identification of Parts

In this section we will try to identify various parts of the system and not only. We will start firstly, by identifying the stakeholders of the system.

2.1 Identyfing Stakeholders

Our stakeholder is Prof. Di Nitto, assigning this project as part of the Software Engineering 2 course. After some analysis, on the lectures and advices we were able to hear during the sessions, we came to the understanding that this is by no means less than a normal industry process of software engineering. This is because we are required to undergo all various stages of the usual process.

The purpose of this assignment is that to prepare us for the outside world, that awaits us. Doing this project, we are sure once we finish it, we will have a good basis upon which we can start facing the work life. We will start by compiling the requirements and specification document (this one) as it is the first in the process. More precisely we will start identifying the various system parts in the following sections. So we want to show that we can identify requirements and specifications, design our web application, implement it and then test it, providing all the documents that developers use in developing real software.

It still remains that we have to focus on some functionalities that the professor gave us to have an application that works. Finally, before going into details, we regard as potential user of our system, everyone, this is because all people participate to events daily, and schedule new ones in the following days.

2.2 Identifying Actors

In this section we are going to identify all possible actors that will interact with our system. Although, the system provides most of the functionalities exclusively to a user that is registered, a guest actor can perform some basic actions described bellow.

- *Guest.* Can explore the website and more importantly, will have the option to register as user to the system and benefit of system's features.

- *User.* A former guest that has undergone the registration process and is by all means part of the system. Therefore, is in possession of virtual credit and has the capabilities to bid an auction, and create them.

- *Administrator.* Person who is responsible for the whole system. In the list of tasks performed by this figure we find the managing of all the users and the maintenance of the web site in general.

2.3 Identifying Scenarios

In the following section we will describe some possible scenarios that may arise in everyday life in people.

- Matteo is a busy man, and he has a pretty much scheduled life and he likes to keep everything by the schedule. Specifically he does the laundry every Saturday afternoon. But something bad happened, his washing machine broke and he needs to replace it. He was considering buying a new one, but those were very expensive. So he tried his luck on the internet. This is when he met GuessBid, a web application which is an innovative system for buying items, based on the idea that the auction will be won not by the highest, but by the lowest bid! Yes, that's true! The only thing is that in order for it to be winning, it needs to be also unique. Matteo was interested by the idea, and he was almost sure if it wins the auction he would win it by a more convenient price, from the stock ones. So he made various bids, in order to beat the competition and he got his washing machine paying much less.

- After the very pleasant experience of the washing machine bought for cheap price on the auction using the GuessBid web application, Matteo thought that he could use it more.In this case he remembered of the old TV he replaced 2 years ago by the innovative LED based one. So he created the auction, and immediately the offers started coming. After the auction was concluded, Matteo earned a reasonable amount of money for something that was occupying place for nothing.

- Finally, after some time looking for a good mobile phone, Matteo finds a newly listed Samsung phone from the latest series. Without waiting too much, Matteo makes a bid, and is really confident in winning the auction. Later on that day, Matteo finds out that he is out bided. That

will not change his mind to bid again on the product, but luckily for him, he spots the chat on the auction site. After reading a bit what the other users wrote about the product, Matteo finds out that the phone is actually for the American Market and works without a SIM card. That depresses Matteo a little bit, but luckily for him that he used the chat, because otherwise he was ready to make another bid.

3 Requirements

Now, we are going the describe the requirements of the system. As we know, the first phase of the process is requirements elicitation, in other words, collecting information from potential users. We simulated this process in our group of three, by interchanging the roles of user and interviewer, and succeeded to define a list of requirements that will apply to a usual user. We will start by describing the ones that involve some function by the actors.

3.1 Functional Requirements

1. *Registering a new user to the system.* The system has to provide a sign up section on its first page.

2. *Login to the system.* The system has to provide a login section for registered users on its first page.

3. *A user can view the listed auctions.* The system will provide a list of auctions sorted by the date (newest first) on its homepage after the user is logged in.By clicking on an auction item from the list the system will redirect to a page where details about the auction will be displayed.

4. *A user can bid in an auction.* After the user logs in, and is redirected to a particular auction, then he/she will be able to place a bid on the aucton.

5. *A user can create his/her own auction.* The system will provide an option for each registered user to create auctions.

6. *A user can chat with other users in the same auction.* The system will provide a chat to all the logged users that are watching a particular auction. (They will be able to chat between each other about the details of the auction).

7. *A user can check the status of his bid.* The system will allow the users to check the status of the bids they have placed on auctions.

8. *Notifications regarding auctions.* The system will send a notification to users that have bid on an auction after it ends telling the user whether they have won the item or not.

9. *History of bidded auctions.* The system will provide users with a history of auctions that have ended, by showing details about each auction.

3.2 Non Functional Requirements

In this part we present the other types of requirements, the non functional one. We include as such the user interface and the documentation that will accompany the whole project.

3.2.1 Interface

GuessBid will be a website application. The main attributes that will characterize our system are: simplicity, compactness and functionality. We set as our main goal, the obligation to guarantee a welcoming feeling to the users once they open the page. Our idea is to provide, a few, main functionalities, which will represent the core of the system, making everything doable in few steps, in other words, we emphasize usability over complexity. In the following pages we provide few mockups of how we see our system after its completion. We begin with the system presentation page.

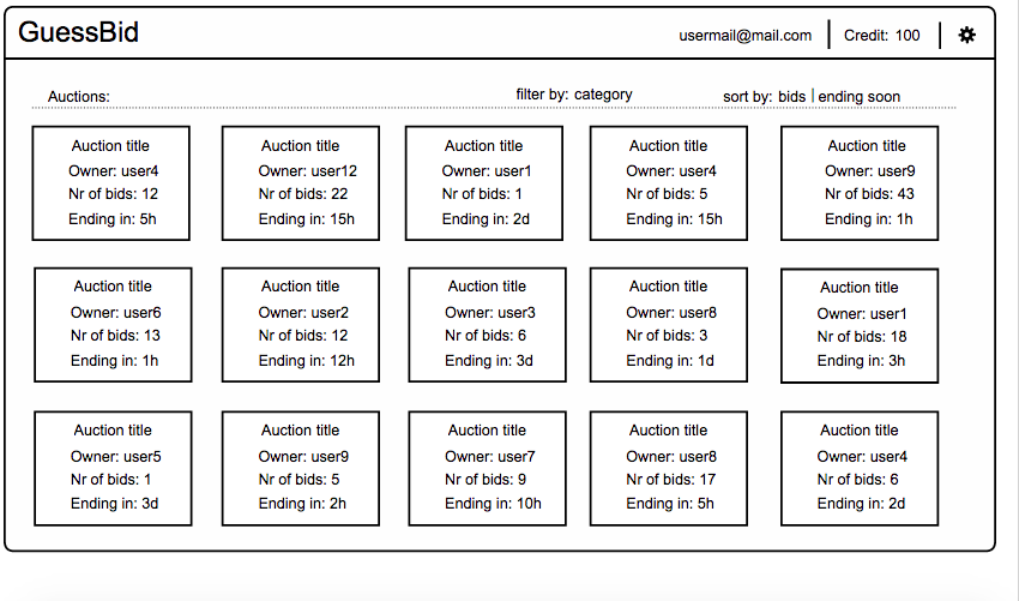

Figure 1: Interface Mockup

Here we provide a mockup of another main screen that will be used most frequently by the user, and it is the one that enables the user to make a bid to some active auction.

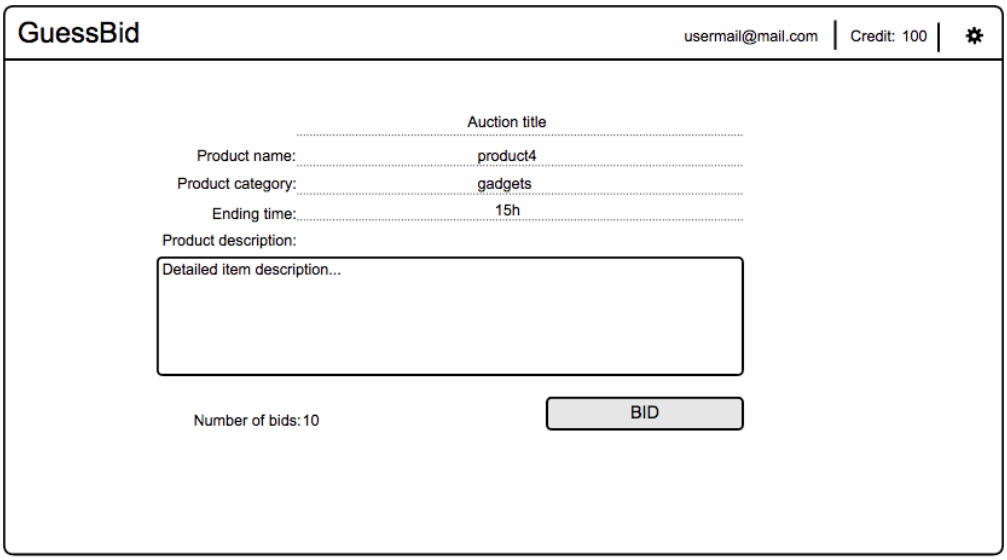

Figure 2: New Bid form

To conclude with, we provide a possible loook of the page that will contain the histroy of alla auctions related to some user, either as owner or bidder.

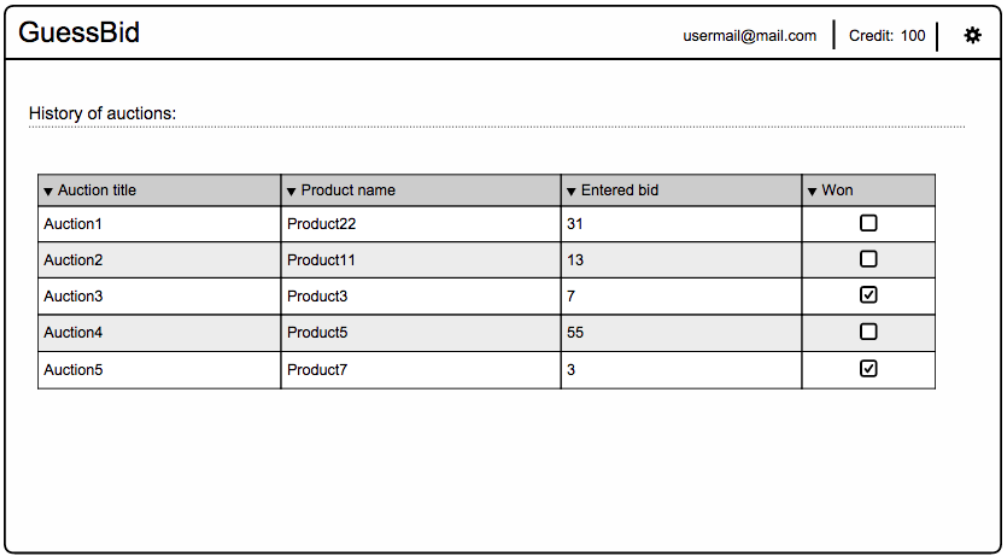

Figure 3: Auction history

3.2.2 Supporting Documentation

We also add as non functional requirement, the whole documentation that will be provided throughout the various phases of the system engineering and development. We will include a series of documents, and following we will provide a short explanation of the purpose of each of them.

Requirement and Specification Document.It is a document which helps to understand what are the exact goals of the system by providing a detailed explanation of the requirements and specifications it will follow. As we saw until now, we can have different types of requirements depending on the fact if they are functional or less. Furthermore, it is possible to find visual flows of many actions that are going to be performed by the system.

Design Document.It is a document that provides the description of the system. It is mostly focused on providing details about the architecture of the software project, hence it attempts to provide all necessary details in regards. The usefulness of it is really noticed when there is a large team that needs to work on the project. In this case it provides the common vision

to all the developers engaged in the process. The ultimate goal of it, is to provide a complete description, while maintaining a high-level view of the software.

Commented Source. We will comment the code for many purposes, the first one being the possibility of code sharing and adaption among teammates. Also this is the best way to help third party people who will in future read our code, understand the meaning in considerably easy way.

Installation Manual. A set of instructions on how to set GuessBid in the machine. This may be trivial in some system, but some others may need Java components to run the web application at its best.

User Manual. The official help document that wil accompany the product. It will have an exausted explanation of all the features of GuessBid as well as a FAQ section, for the most known questions that may be asked.

Testing Document. This document will be provided after testing stage. Where we will exchange our products with some other team, and perform acceptance tests on their products, and viceversa. A must for this one, is high objectivity in every aspect of the evaluation process.

4 Models

Following we present the use case scenarios that were easy to derive using the knowledge acquired in the previous sections.

4.1 Use Cases

Name	Registering as new user
Actors	User
Conditions	No conditions required to create an account.
Flow of Events	• The user enters the home page of the system.The system shows a particular section for new users. • The user hits the button to sign up.The system provides a blank form for registration. • The user fills the fields and clicks Proceed button.
Exit Conditions	The system stores all information about the new user.
Exceptions	• If the user leaves blank some of the required fields in the registration form, the system fill show a warning message. • The user mistakes the newly created password, so the creation of the account cannot continue.

Name	Login
Actors	User or administrator.
Conditions	The user has successfully register in the account previously with an e-mail address that was verified by the system.
Flow of Events	• The user/administrator enters the home page of the system. • The system shows the page and the section for log in. • The user/administrator enters his e-mail address and password, and clicks log in-button to proceed. • The system shows the profile of the user/administrator.
Exit Conditions	There are no exit conditions.
Exceptions	If the information that the user/administrator inserted in the fields are not correct, a message with a possible errors is shown to the user/administrator.

Name	Create an auction
Actors	User.
Conditions	The user has to be logged in.
Flow of Events	• The user clicks on the create new auction button. • The system shows a page with sections that need to be fulfilled so the auction can be valid and later on published on the site. • The user provides information about the auction: Description of the product that the user wants to put on auction, selects a category to which the product belongs, sets expiration date after which the auction expires. Next the user proceeds to save the newly listed auction. • The system automatically updates the list with auctions with the newly listed one.
Exit Conditions	The home page of the user is now reloaded containing the auction he/she posted.
Exceptions	If the form of the auction is not adequately fulfilled the system will show a warning message to the user.

Name	Listing Auctions
Actors	User/Adminstrator.
Conditions	The user/administrator has to be logged in into the system.
Flow of Events	• After logging into the system, the first page that the user/administrator can see is the home page presenting auctions listed recently. All auctions show description of the product, name of the user that listed the auction, time when it was listed, and the date when it expires. • To filter the auctions by a category the user can select that option from the menu. • After pressing the wanted category, the system will show to the user all auctions that were listed to that specific category, starting from the most recent one. • If the user wants to find out more details about an auction, clicks on the auction. • The system provides information about the product listed.
Exit Conditions	Home page is reloaded with the most recently listed auctions (not filtered by a category).
Exceptions	No exceptions in this case.

Name	Bids on Auction
Actors	User.
Conditions	The user has to be logged in and to be on the specific auction site (he/she is interested) where all the details about the auction are present.
Flow of Events	• The user can enter his/her bid to the section provided, and by clicking the Bid button. • The system will automatically decrease users account for the 2 credits (which is cost per bid). • The system informs the user about his/hers current position of the bidding.
Exit Conditions	Home page is reloaded with the most recently listed auctions, and the auctions state on which the user bided is saved in the database.
Exceptions	No exceptions in this case.

Name	Auction chat.
Actors	User.
Conditions	The user has to be logged in into the system and to be on the specific auction site where he/she already made a bid.
Flow of Events	• When the user enter to the site of the auction, the system shows a chat where users currently bidding on the product exchange messages. • The user writes his/hers message in the section provided by the system, and presses the button to send the message. • The system shows the message to the other users on the chat.
Exit Conditions	The home page with auctions listed is reloaded, and the chat on the auction site is saved in the systems database.
Exceptions	No exceptions in this case.

Name	Auction History.
Actors	User.
Conditions	The user has to be logged in.
Flow of Events	• The user clicks on the button for Auctions history on his home page. • The system provides the user list of auctions the user bided on. • The user clicks on a particular auction he/she is interested to know more about. • The system shows description of the product listed in the auction, time when the product was listed, and the position on which the user finished the auction.
Exit Conditions	The homepage with auctions listed is reloaded.
Exceptions	None.

Name	Auction Notifications.
Actors	User.
Conditions	The user has to be logged in into the system and it must be either a participant of an auction, or owner of an auction.
Flow of Events	• After the auction time expired, the system will show the user the outcome of the auction (for bidder-final position in the auction or if seller-amount paid for the product) with description of the product that was listed on the auction (because the user can bid or sell more auctions at the same time).
Exit Conditions	The home page with auctions is reloaded. If the user wins an auction credit will be decreased from his/hers account. If the user was the owner of the auction credit will be added to his/hers account.
Exceptions	No exceptions.

4.2 UML diagrams

In this section we will provide a conceptual representation of the more important use cases listed previously. We will combine the ones that perform alike functions, so that we avoid lot of repetitions. We decided to split the diagrams from the actual use cases to emphasize the point that they are not the same thing, as often are confused like so. They do represent the same set of actions, but in a completely different way. As the use cases were previously described in details we will just show the diagrams. We will start by providing an overview diagram that depicts the functions that the system offers to the user.

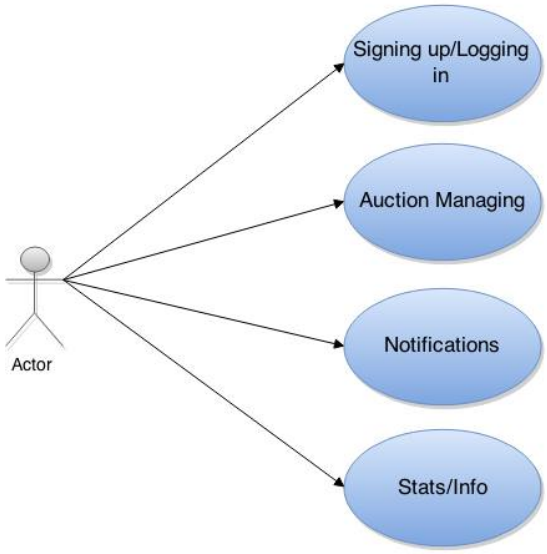

Figure 4: System Diagram

4.2.1 Auction Managing

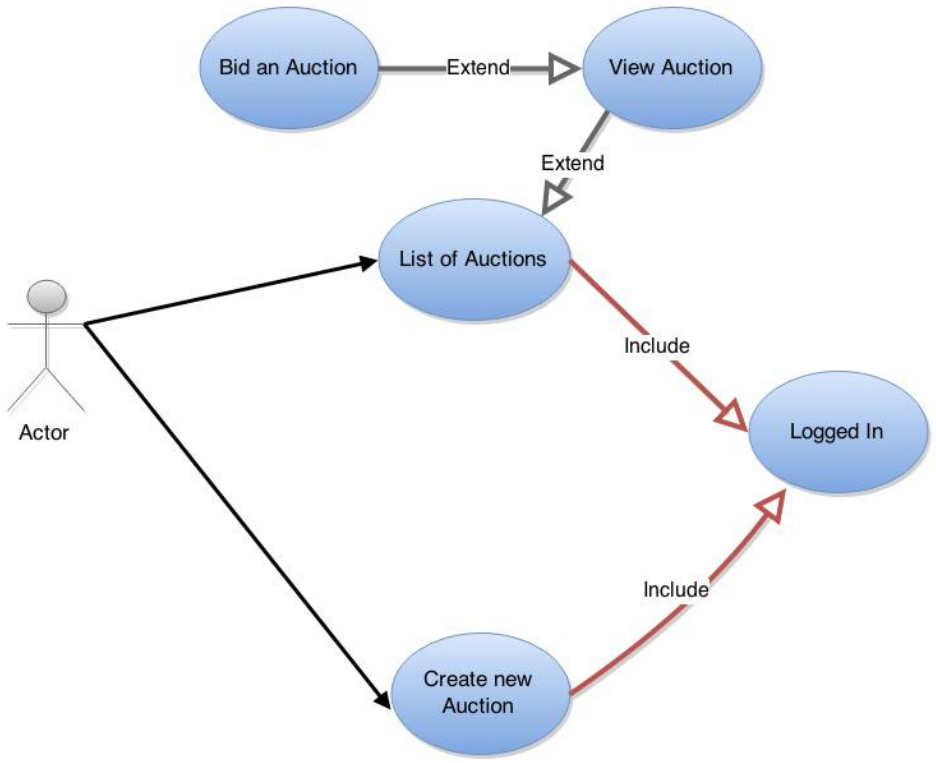

Figure 5: Auction managing process.

4.2.2 Auction Statistics

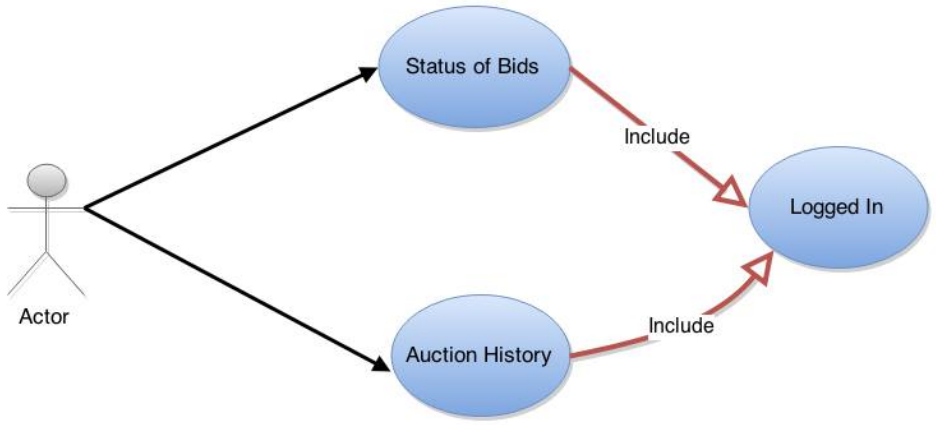

Figure 6: User accesses the statistics of auctions.

4.2.3 Login and SignUp

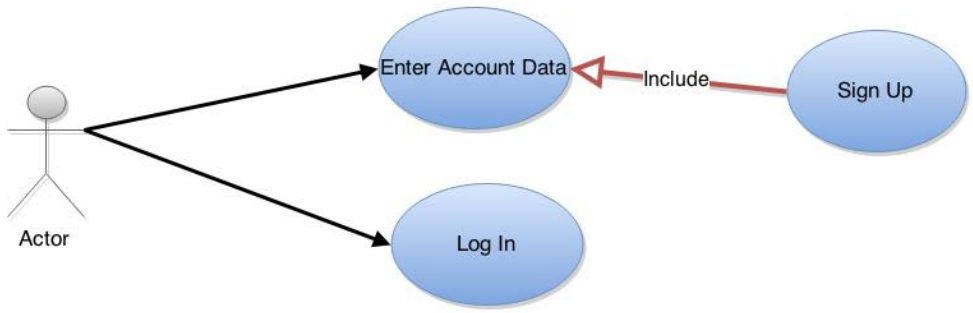

Figure 7: Login and signup diagram.

4.2.4 Notification System

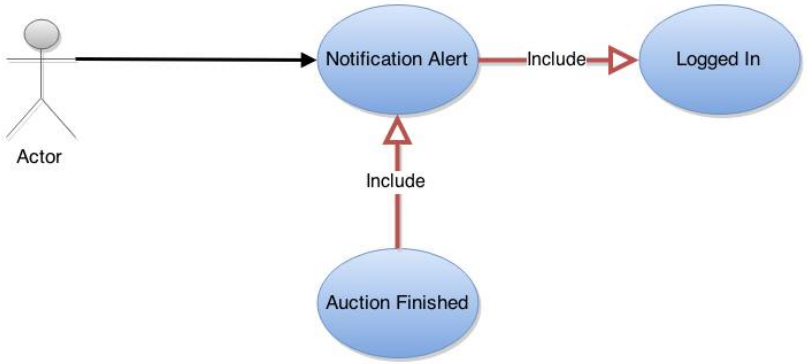

Figure 8: Notification for auctions.

4.3 Sequence Diagrams

4.3.1 Sign Up

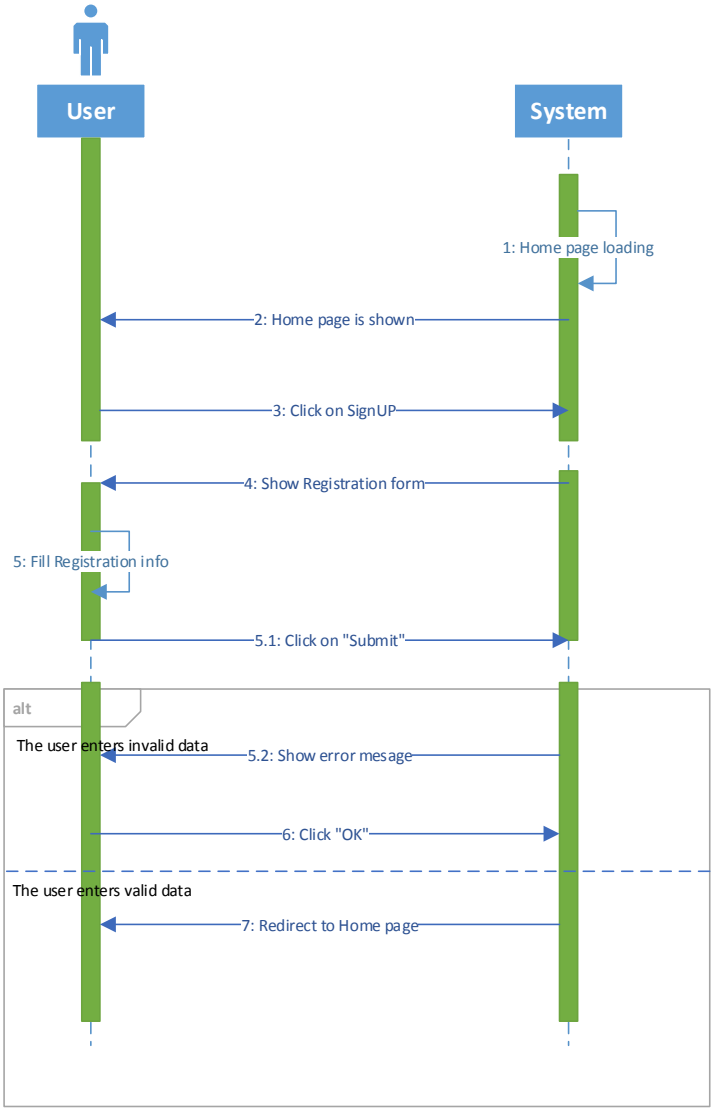

Figure 9: Sign up sequence.

4.3.2 Log in

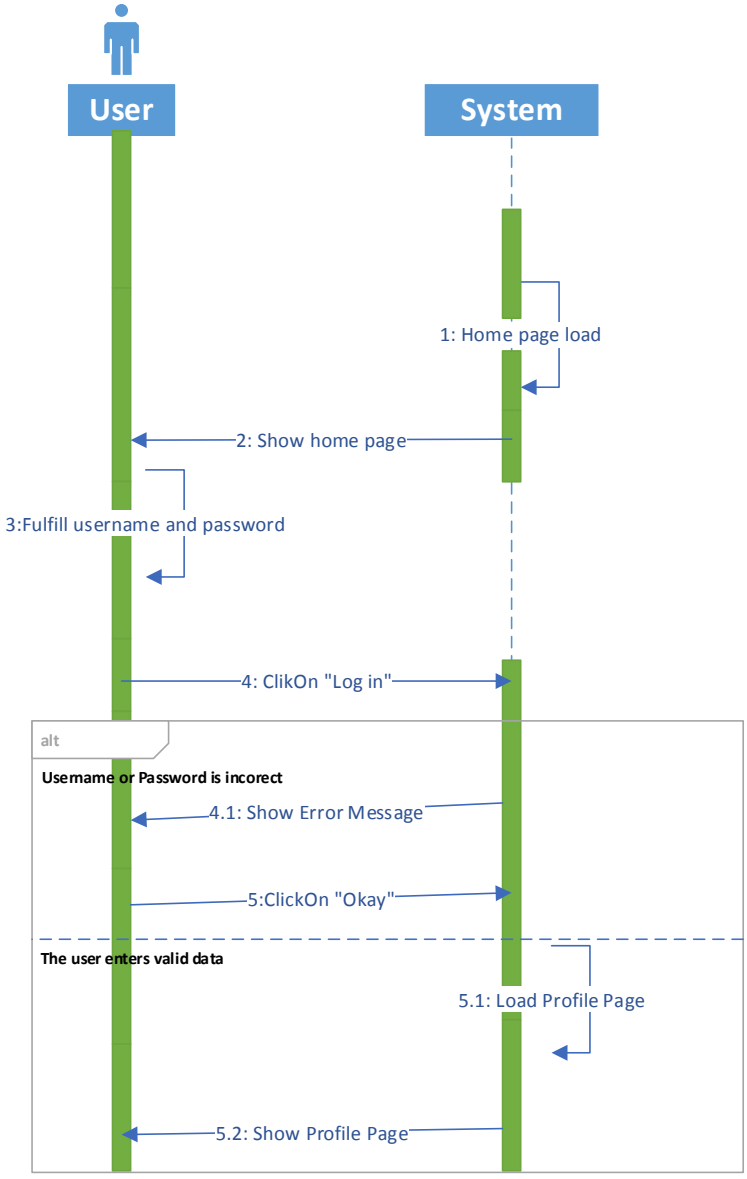

Figure 10: Log in sequence.

4.3.3 New Auction

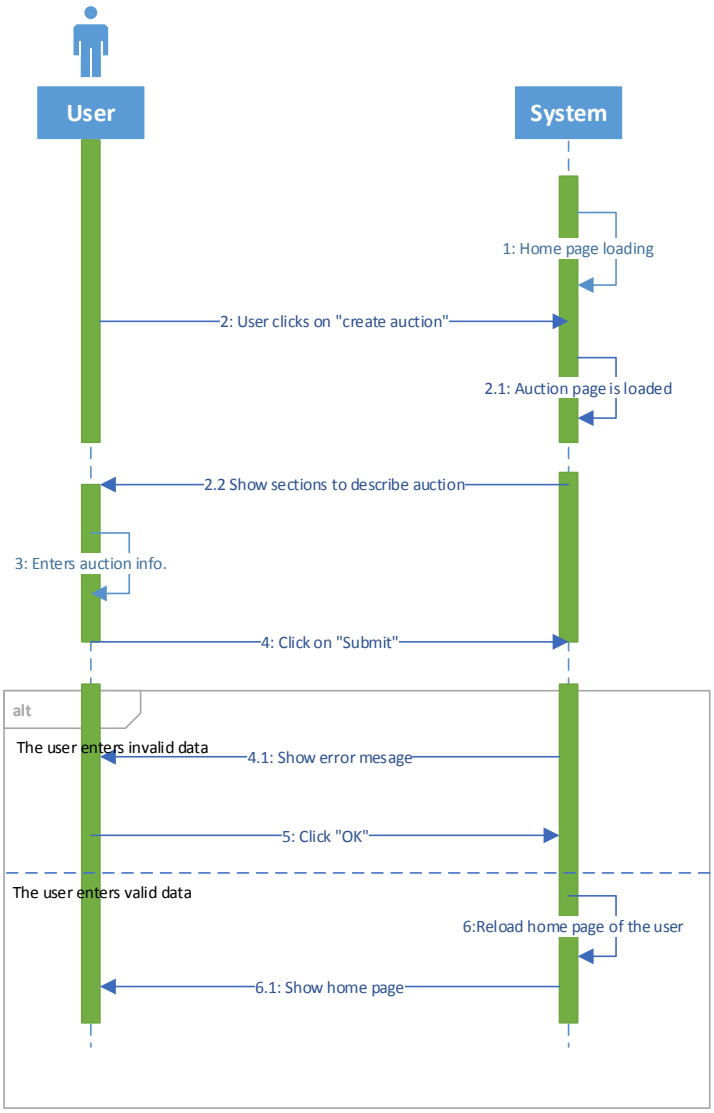

Figure 11: Creation of new auction.

4.3.4 Auction Bid

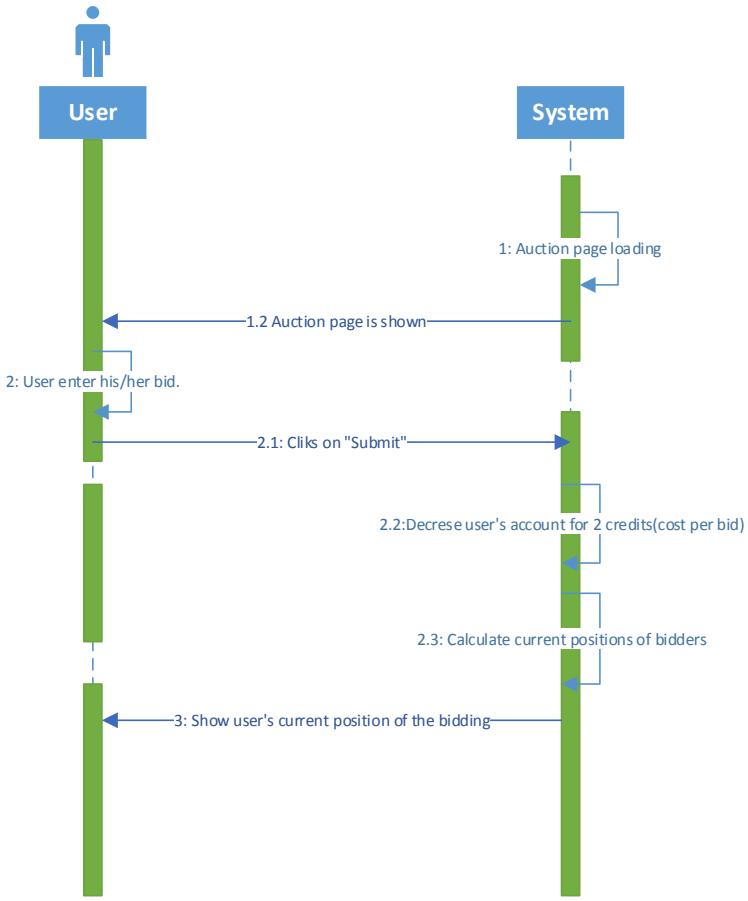

Figure 12: Auction bidding process.

4.3.5 Auction Chat

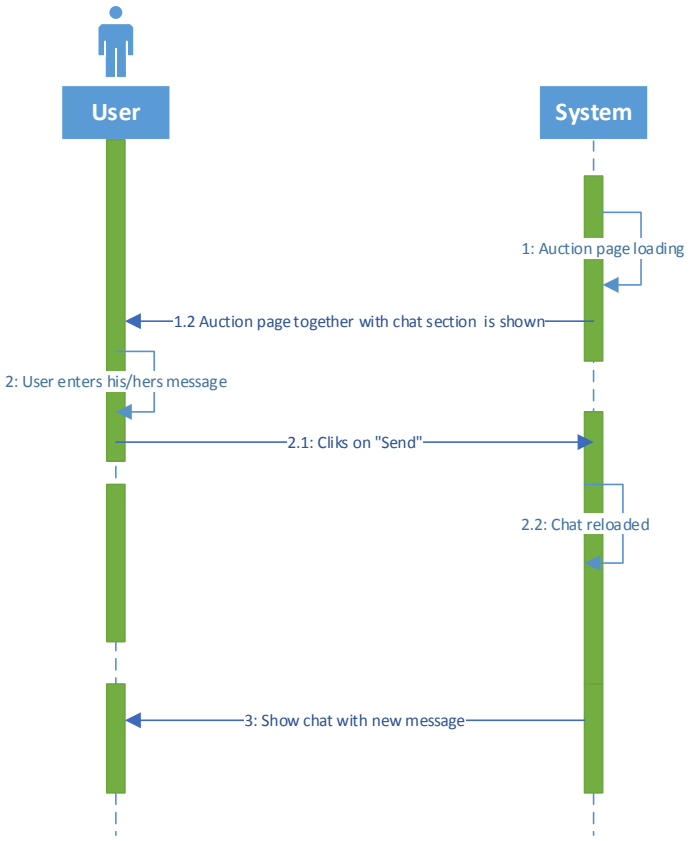

Figure 13: Auction-specific chat.

4.3.6 Auction List

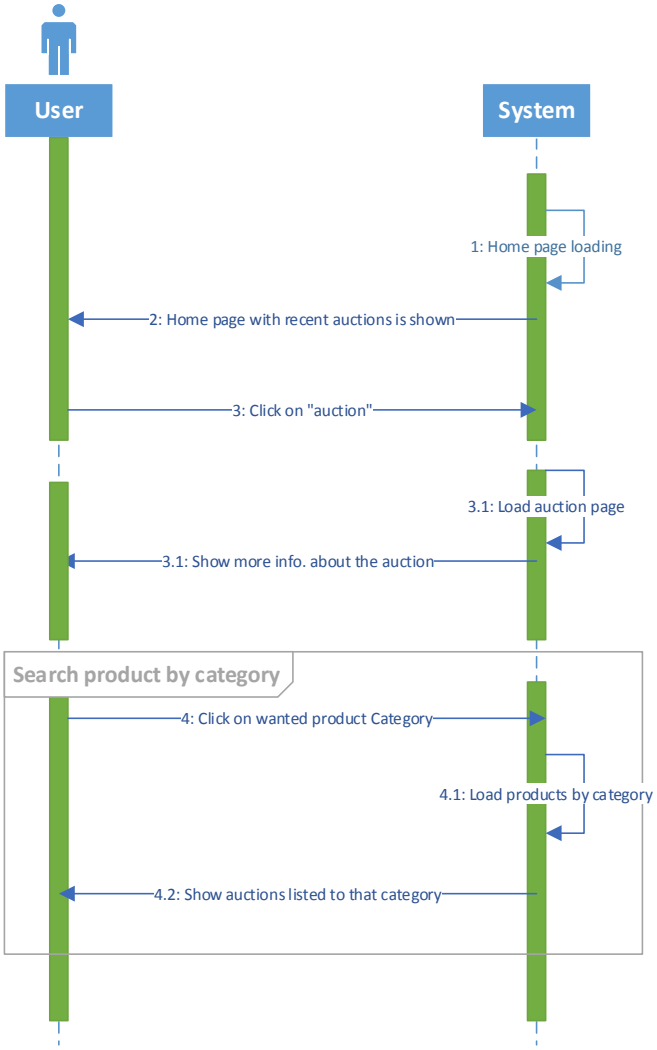

Figure 14: Process of getting the list of active auctions.

4.3.7 Auction History

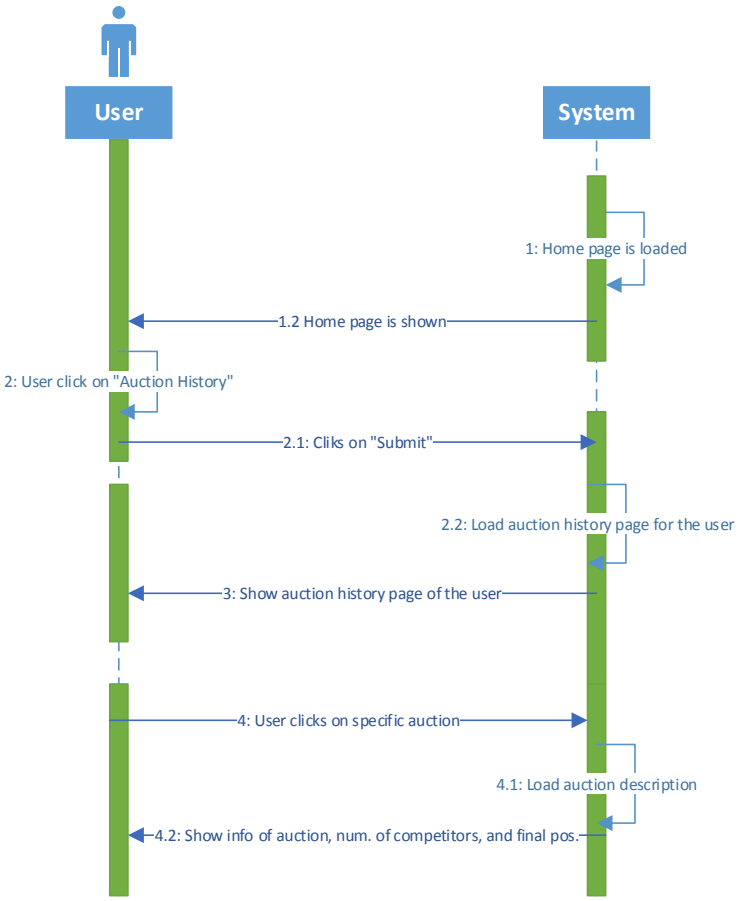

Figure 15: List the concluded auctions.

4.3.8 Auction Notifications

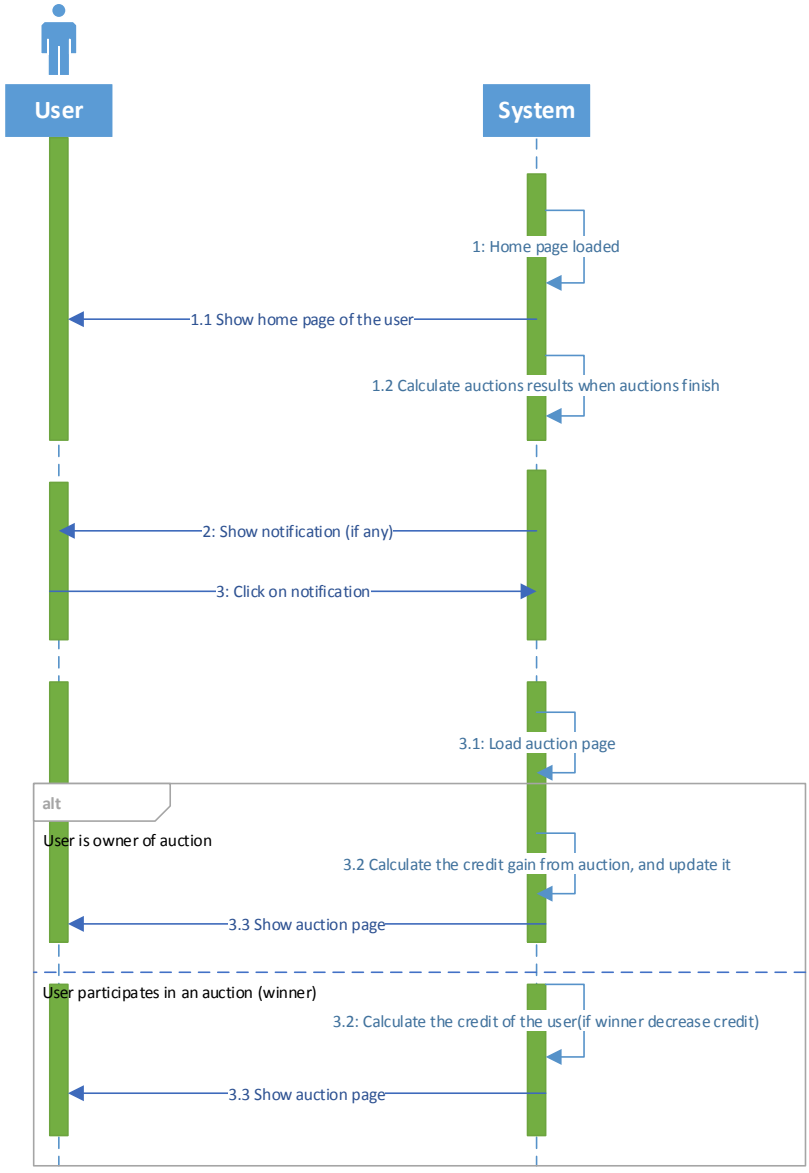

Figure 16: Auction notifications procedure.

4.4 Class Diagram

It is time to present the class diagram. We decided to do it last, because we needed a clear definition of all the relations and functions that the entities may have.

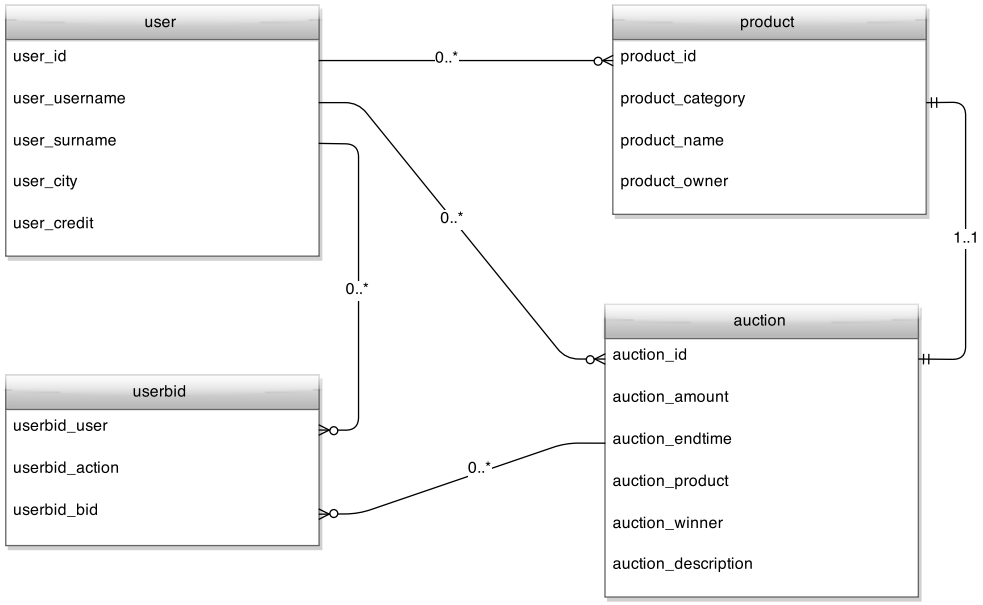

Figure 17: The system class diagram.

4.5 Alloy Model

To finish with, in this section we will provide a model of the system in the Alloy modeling tool. The model attempts to represent the world of our system, as described on the class diagram previously, questioning the fact if it can be consistent or no. By consistency we mean, that given a set of signatures and facts among them, we will use some assertions to find out whether or not our world is consisted, which would mean that the analyzer didn't find a counter example. Of course, as the power of alloy is limited, we will represent the system with several levels of abstractions, because it could be very complex, if not impossible, to model everything. First we post a possible model that was obtained in alloy, followed by the complete code with comments, and finally the execution screenshot with some assertions.

4.5.1 Alloy World

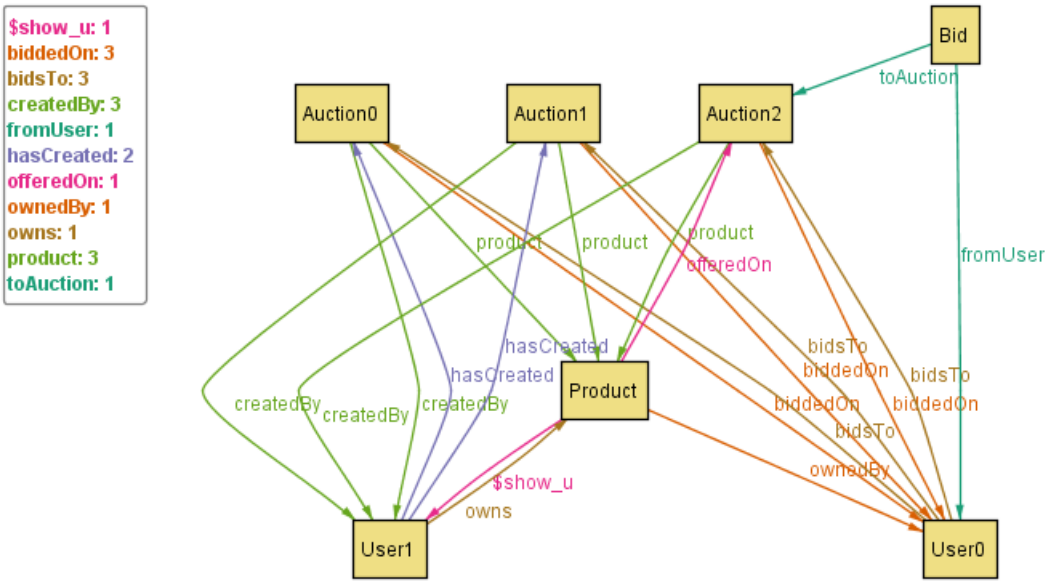

Figure 18: Consistent world generated by alloy.

4.5.2 Code

```
module GuessBId

//Signatures
sig User{
   hasCreated :set Auction,
   owns     : set Product,
   makes   : set Bid,
   bidsTo   : set Auction
}

sig Bid{
   toAuction: one Auction,
   fromUser: one User
}

sig Product{
   ownedBy : one User,
   offeredOn: one Auction
}

sig Auction{
   createdBy: one User,
   biddedOn :set User,
   product: one Product
}

//Facts
fact ProductAndUser{

   //Two users can't, simultaneously, own the same product
   all disj u,v :User | u.owns & v.owns =none

   //Every product must be owned by some user
   all p : Product| some u :User | p in u.owns

}

//User, Bid, Auction constraints
```

```
fact UserAuctionBid{

    all u : User | u.hasCreated & u.bidsTo =none

   all u : User | all a: Auction |
           a not in u.bidsTo implies u not in a.biddedOn

   all u : User | all a :Auction |
           u in a.createdBy implies not(a in u.bidsTo)

   all u : User | all a :Auction |
           a in u.hasCreated implies u not in a.biddedOn

   all b : Bid | all a :Auction |
           a.createdBy & b.fromUser =none

}

//Assertion for users not bidding on theirown auctions
assert NoSelfBid{
   all u : User | all a :Auction |
           a not in u.bidsTo implies u not in a.biddedOn
}

//All products must belong to someone
assert AllProductsHaveOwner{
   all p : Product | some u :User |
           p in u.owns
}

//Every product can have a unique owner
assert EveryProductHasUniqueOwner{
   all disj u, v :User |
           u.owns & v.owns =none
}

check NoSelfBid
check AllProductsHaveOwner
check EveryProductHasUniqueOwner
```

```
pred show{

}

run show for 3
```

4.5.3 Alloy Execution Result

4 commands were executed. The results are:
#1: No counterexample found. NoSelfBid may be valid.
#2: No counterexample found. AllProductsHaveOwner may be valid.
#3: No counterexample found. EveryProductHasUniqueOwner may be valid.
#4: **Instance found.** show is consistent.

Figure 19: Picture of the execution result.

5 Used Tools

1. LaTeX
2. Visio 2013
3. Alloy
4. Creately
5. moqups.com