# PlzDontSue, LLC

**eBamazon**
**Phase 2: Design Report**
**for eBusiness Platform**

**Version 1.3**

**Team #11 Members:**
Marissa Almoslino
Jacob Luceno
Zhandaulet Saduakas
Brian Thibeault

# Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 04/16/2019 | 1.0 | Design of Use-case scenarios diagrams (sequence and petri-net) | ALST |
| 04/17/2019 | 1.1 | Added E/R and collaboration diagrams for the entire system | ALST |
| 04/18/2019 | 1.2 | Worked on the pseudocode and the GUI screens | ALST |
| 04/19/2019 | 1.3 | Updated E/R and collaboration diagram, put all the the files in Git repo and added minutes of group meetings. Also, finalized the report | ALST |

# Table of Contents

# Design Report

## 1. Introduction

This section includes a description of the purpose of the Design Report (DR) and the scope of an eBusiness platform called eBamazon. Additionally, a list of terms is provided along with an a Collaboration Class Diagram.

### 1.1 Purpose

This Design Report serves as a detailed design specification for all subsystems of eBamazon, complete with a collaboration class diagram; scenarios, sequence diagrams, and Petri-nets for each use case; an E/R diagram for the entire system; detailed pseudocode for every class method; and wireframes for all GUI screens.

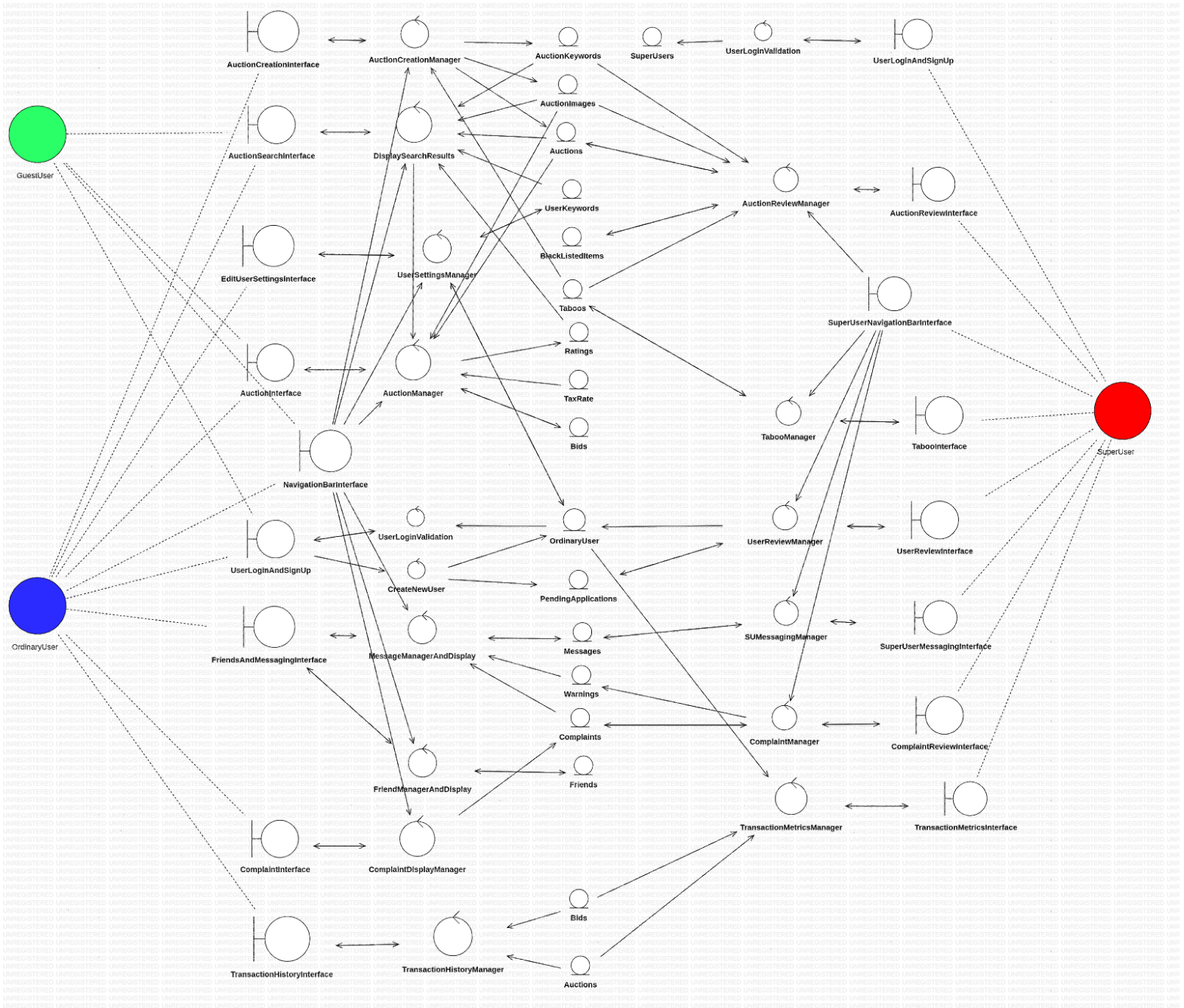
### 1.2 Scope

eBamazon is intended to be a convenient and easy-to-use application for users trying to bid on, buy, or sell user-defined items. The platform is supported by a database with its system management and specification functions. We will have a database with a local server that will store all items for sale and users (sorted by their type). We would like to provide a simple user experience with the most appropriate design.

### 1.3 Definitions, Acronyms, and Abbreviations

GU : Guest User
OU : Ordinary User
SU : Super User
DB : Database
GUI : Graphical User Interface
SRS : Software Requirements Specification
DR: Design Report

## 1.4 Collaboration Class Diagram



# 2. Use-Case Scenarios and Diagrams

This section includes normal and, in some instances, exceptional scenarios for each use case. Moreover, each use case will be accompanied by a Sequence Class Diagram as well as a Petri-net, both of which will help visualize the functionalities of each use case. The pre-conditions laid out in the SRS still apply (but are not provided here).

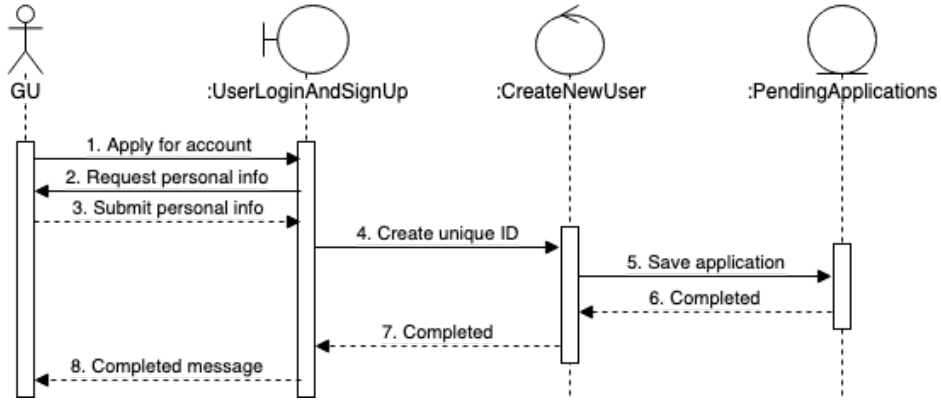
**Use Case 1:** 'Apply for an Account'

**Normal Scenario:**

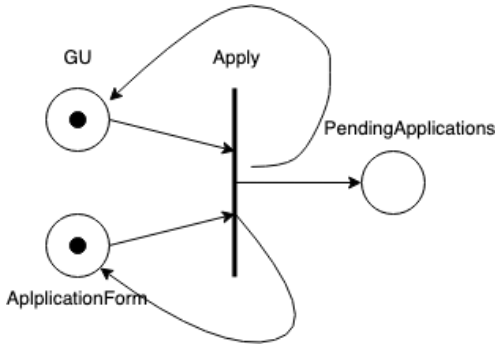
1. GU clicks on 'Apply'.
2. GU completes an online request for information including Name, Username, Address, Phone Number, and Credit Card Number.
3. The system creates a unique ID which is hidden from GU but is included in the application.
4. The system places the completed application in a queue for a SU to review.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



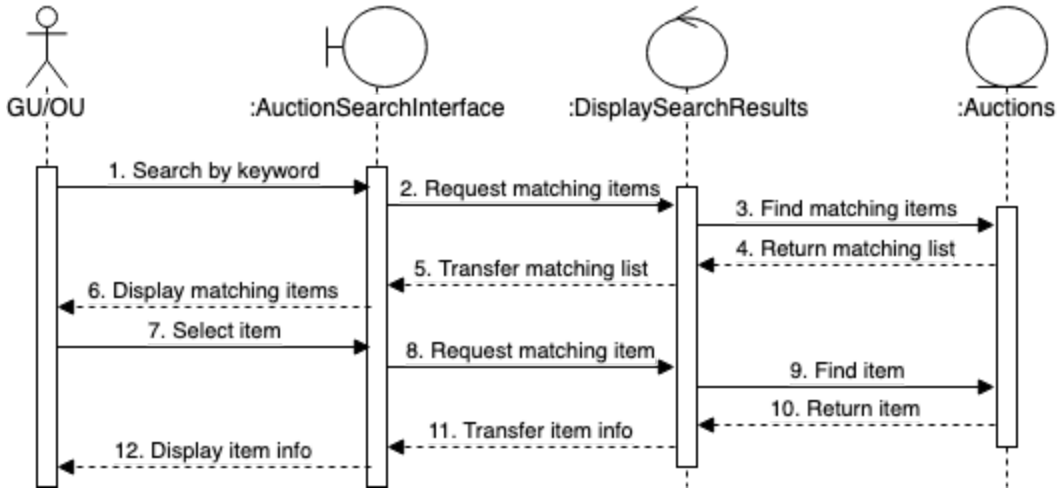

**Use Case 2:** 'Search/Browse Available Items'

**Normal Scenario:**

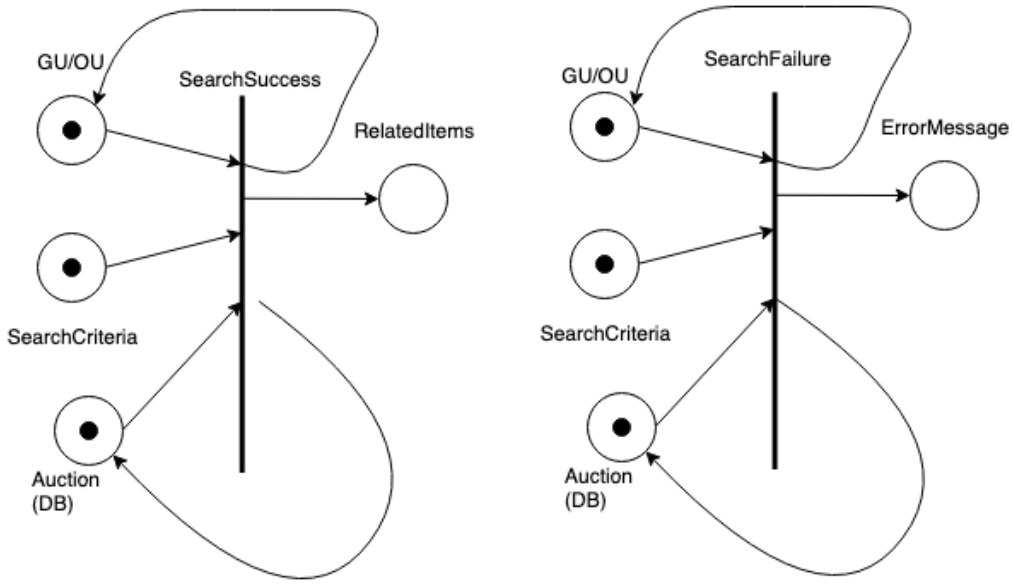
1. The user searches for the keyword “table” using the search bar at the top of the screen.
2. The user’s page is populated with items for sale matching the given keyword.
3. The user is free to click around the items based on relevance and price.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



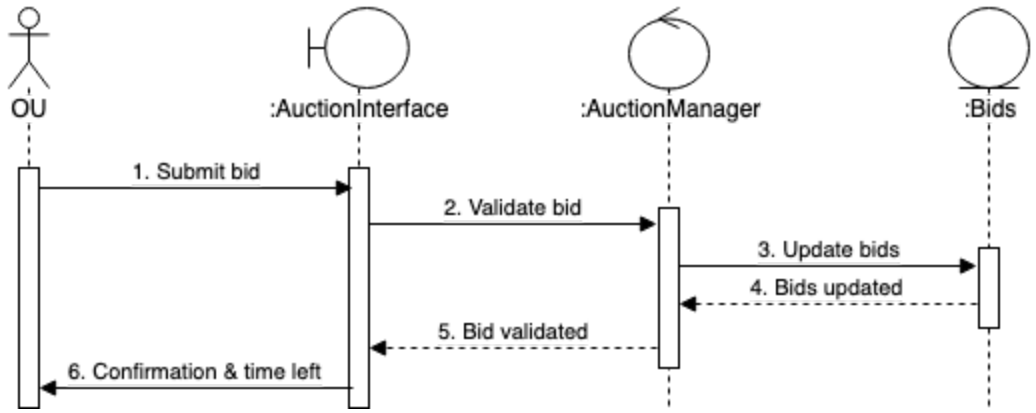
**Use Case 3:** 'Bid on an Available Item'

**Normal Scenario:**

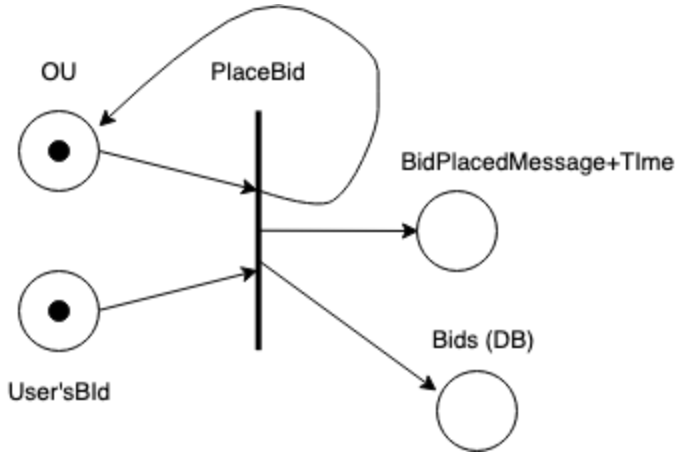
1. The user finds a table they'd like to bid on.
2. The auction for the table is currently ongoing.
3. The user places their specified bid in the dialog box prompting for it.
4. They are sent a message stating their bid has been received and remaining time in auction.
5. At the conclusion of the auction, the user's credit card on file is charged, and the OU hosting the item for sale is sent the funds. The OU hosting the item is also sent the information for the user who bought the item which is required to deliver the item to that user.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



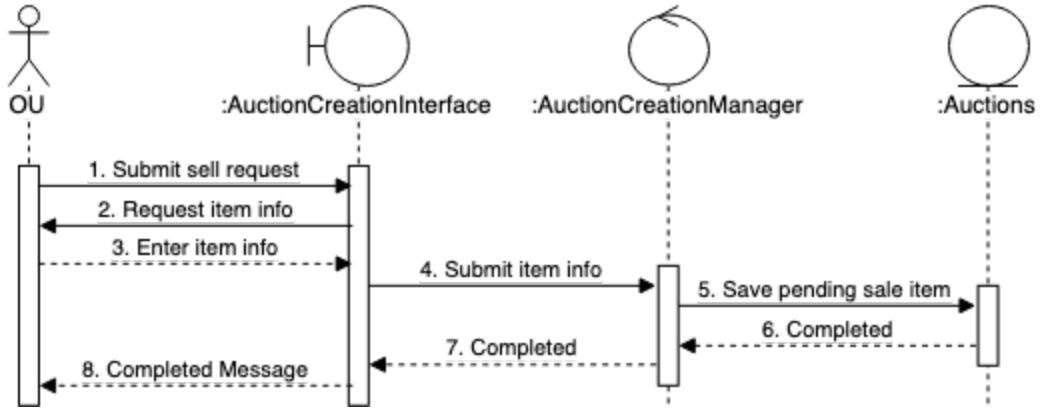
**Use Case 4:** 'Submit a Request To Sell'

**Normal Scenario:**

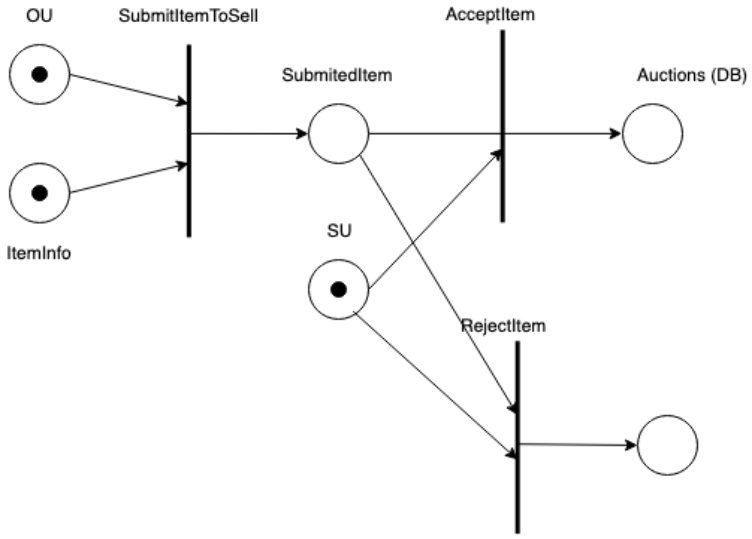
1. An OU has a table they wish to sell.
2. They log onto their account and submit the form required to list the item.
3. An SU will review the form and authorize it for sale, which will add it to the database containing items currently for sale.
4. The item is now searchable by other users based on the keyword information provided.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



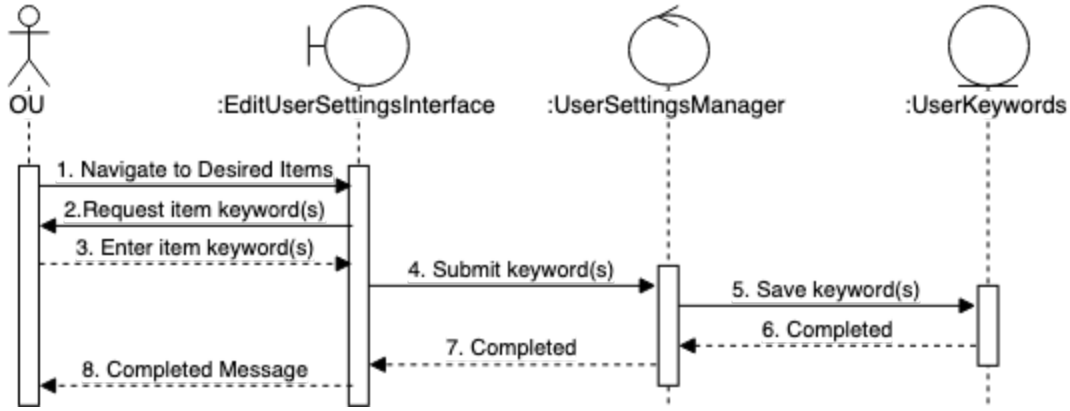
**Use Case 5:** 'Submit Purchasing Intentions'

**Normal Scenario:**

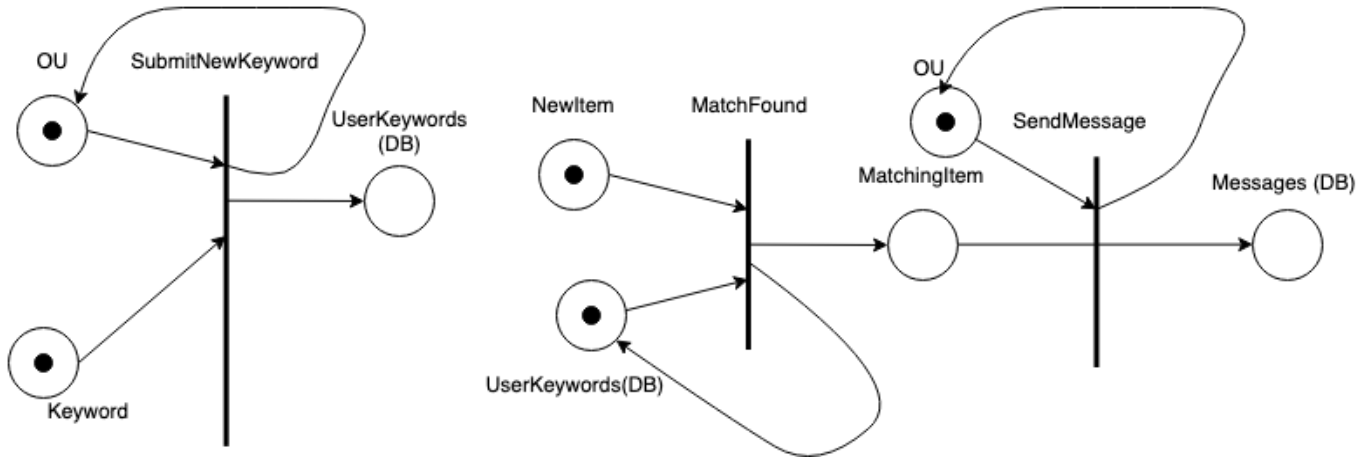
1. The user wishes to purchase chairs for their new table but none are currently for sale.
2. The user navigates to their "desired items" list and may enter keyword "chairs".
3. When the system detects an item matching that keyword has been listed for sale, the user looking for the item will receive a notification in their site inbox directing them to the relevant auction.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



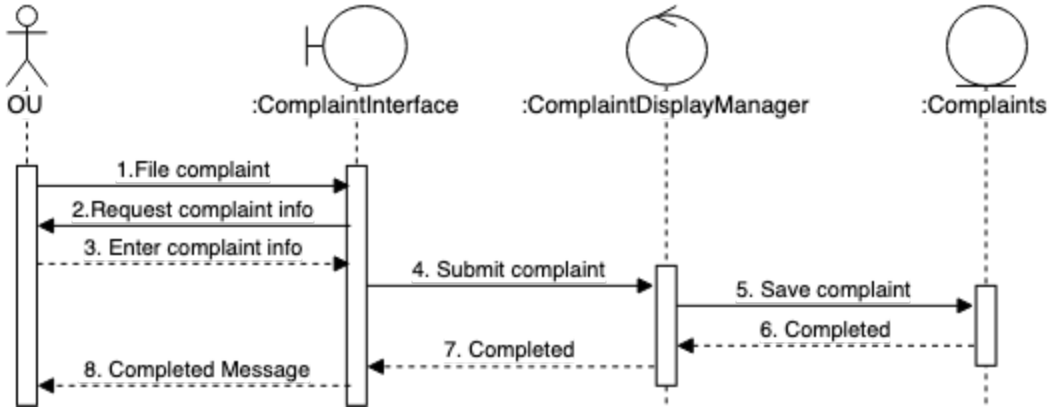
**Use Case 6:** 'File Complaint'

**Normal Scenario:**

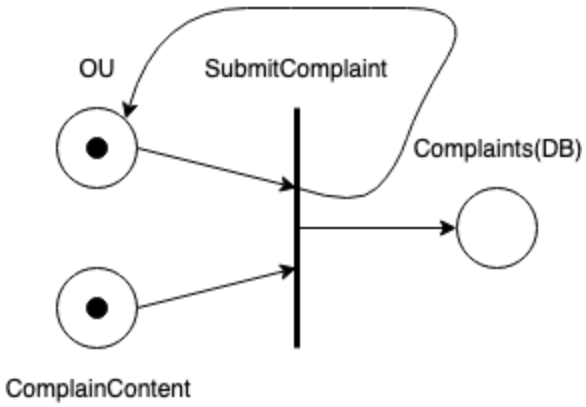
1. The user has won a bid for a table they wanted.
2. The user waits the amount of time the buyer specified but their item never arrives.
3. The user submits a complaint to an authorised SU from the "submit a complaint" page and the SU will respond via the applications messaging system in order to arbitrate.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



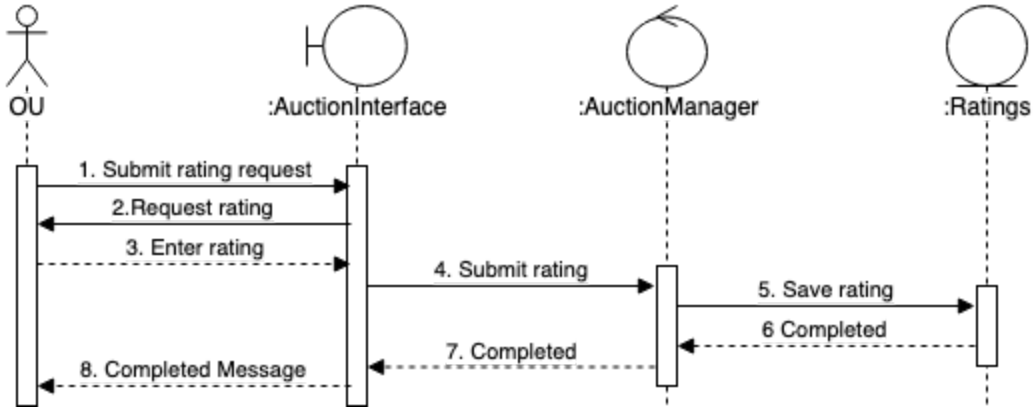
**Use Case 7:** 'Grade Another User'

**Normal Scenario:**

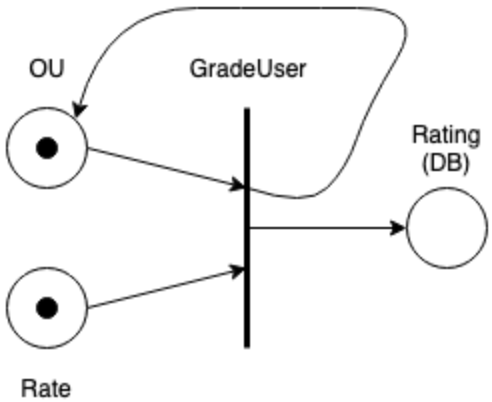
1. The chairs a user has purchased arrive, but in slightly worse condition than the user expected based on the pictures of the item.
2. The user is prompted by the system to grade the seller.
3. The user rates the seller 3 stars since the item did arrive but not as described.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



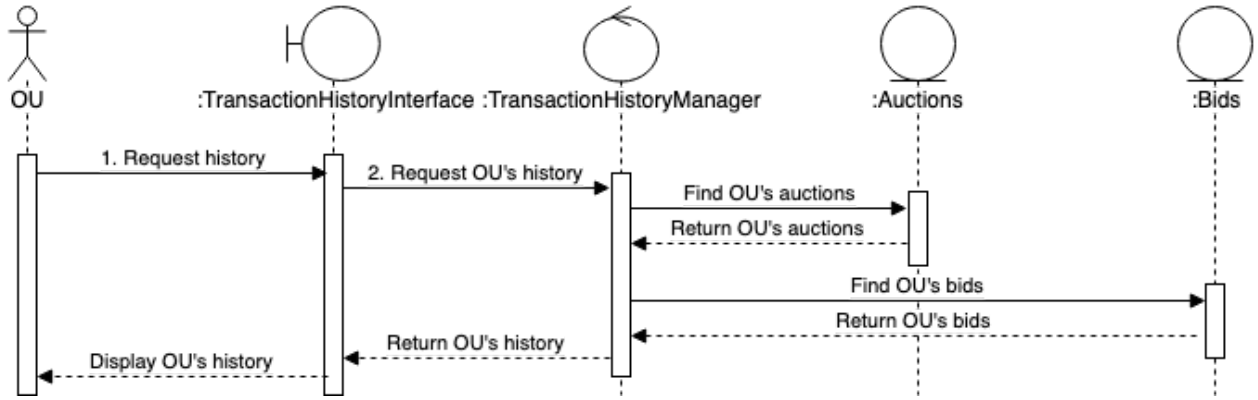
**Use Case 8:** 'View Transaction History'

**Normal Scenario:**

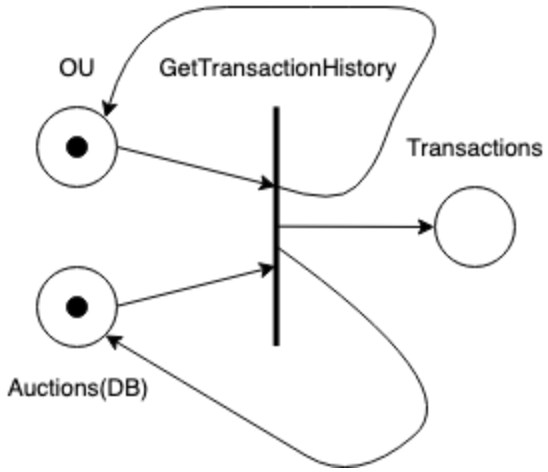
1. The OU navigates to the "Transaction History" page.
2. The page is populated with all transactions that user has participated in.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



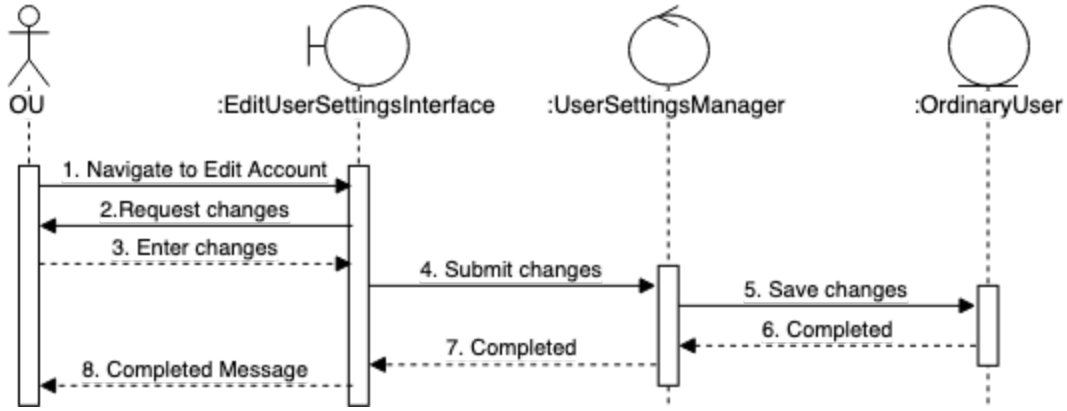
**Use Case 9:** 'Edit Account Information"

**Normal Scenario:**

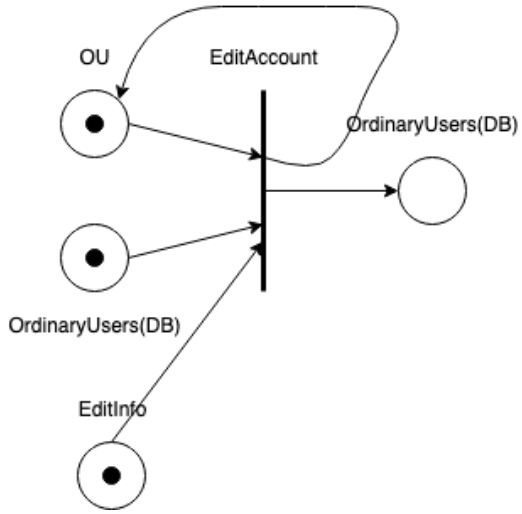
1. The user navigates to the "Edit Account" page.
2. They decide to alter their password.
3. They click the "Edit Password" button.
4. They are prompted for their old password, and a new password, as well as to retype their new password.
5. Their password is updated in the database.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



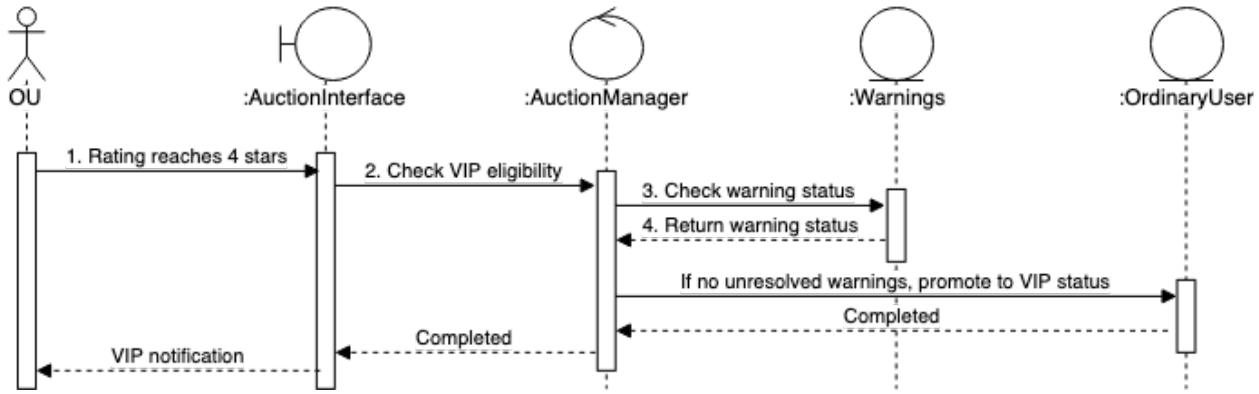
**Use Case 10:** 'Be Promoted to VIP'

**Normal Scenario:**

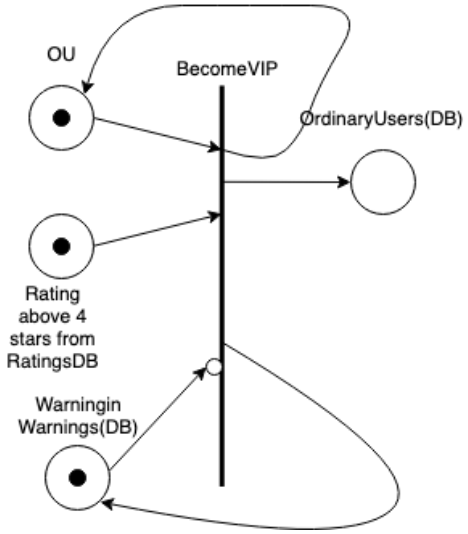
1. An OU has just sold any item and their review brings them above 4 stars.
2. Since there are no unresolved warnings associated with their profile, they automatically become a VIP member.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



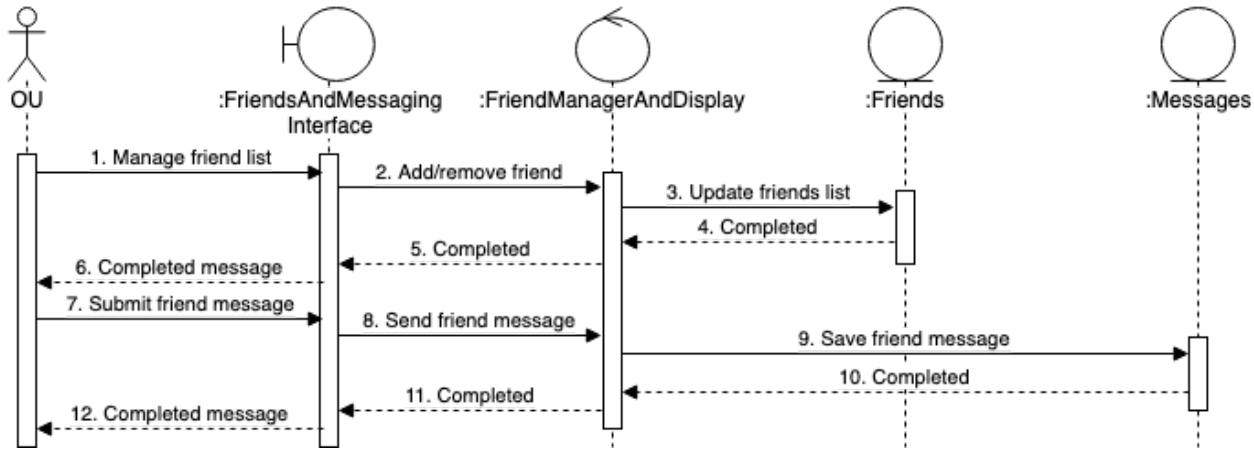
**Use Case 11:** ‘Maintain Friends’

**Normal Scenario:**

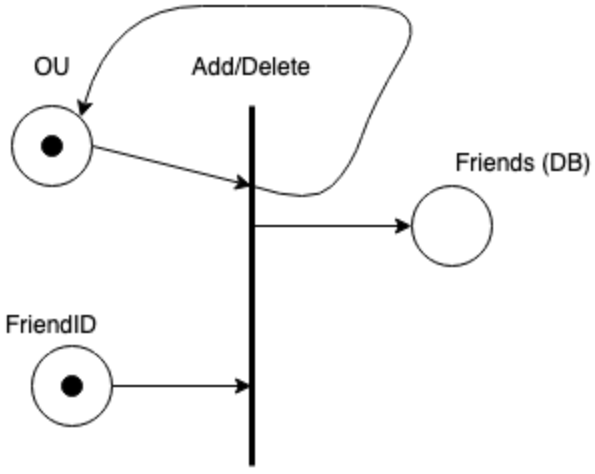
1. The OU navigates to the “Manage Friend List” page where they can add/delete friends.
2. They add the newest friend they’ve made through a recent sale.
3. They send a friends-only message and the member just added to the friends list receives it.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



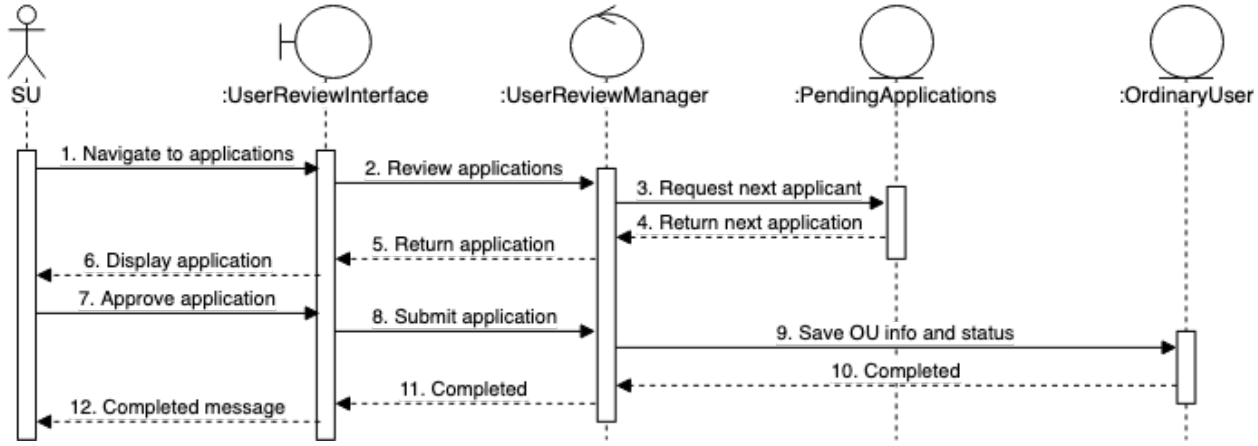
**Use Case 12:** 'Process Applications'

**Normal Scenario:**

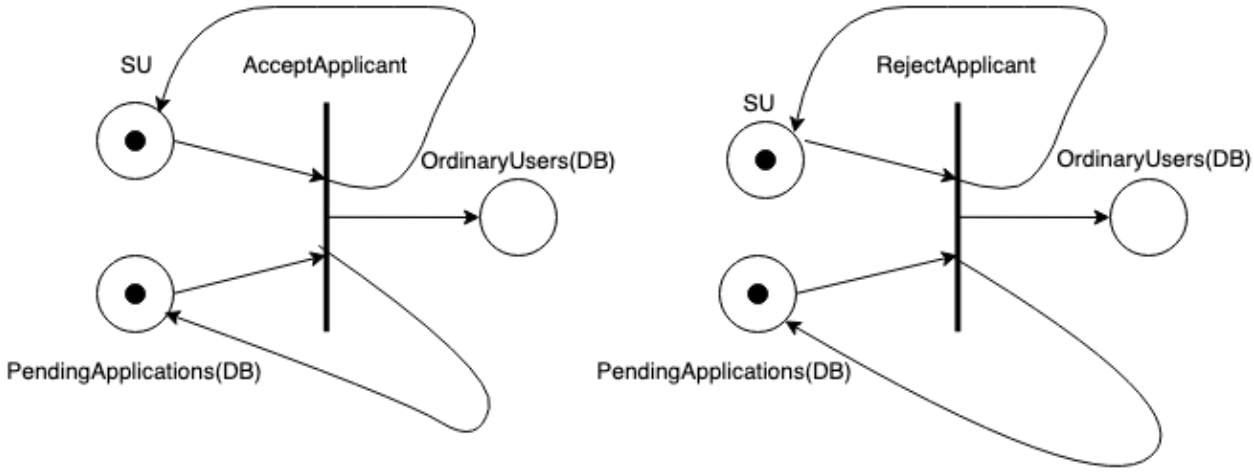
1. The SU can load the first applicant in the queue.
2. They review their info and validate the card information.
3. They accept the application.
4. The OU's account is created with the specified username, and next time the user logs in they are prompted to set their password.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



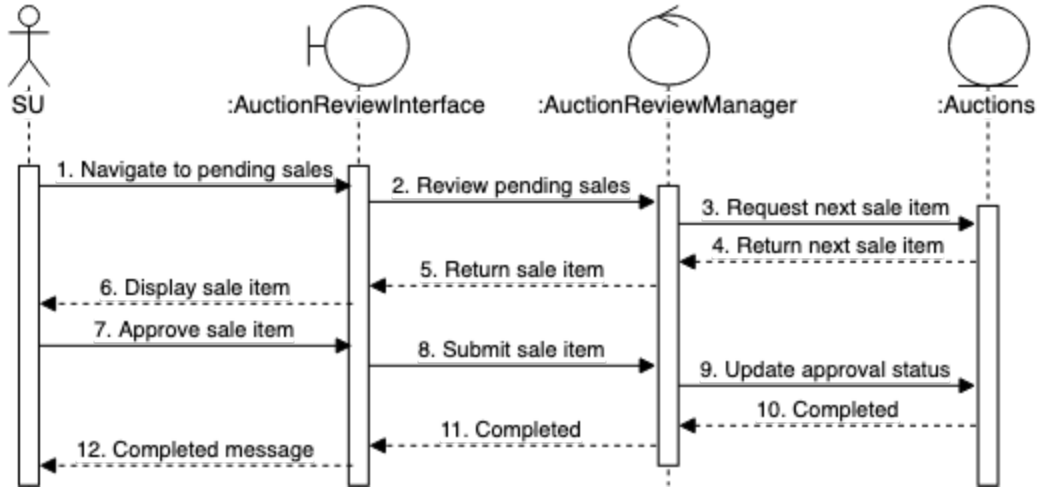
**Use Case 13:** 'Review Sale'

**Normal Scenario:**

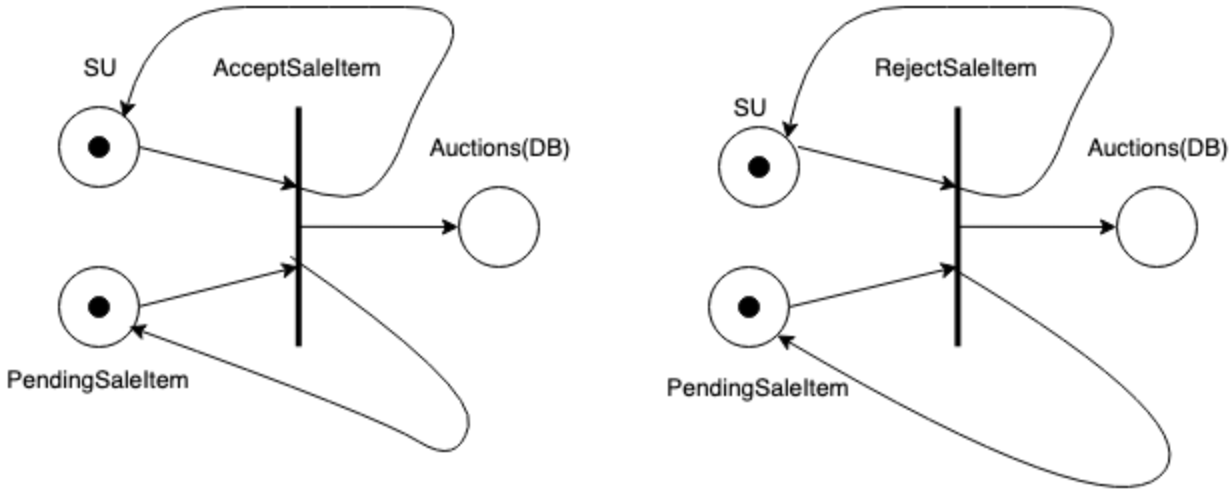
1. The SU is notified of a new sale.
2. They review the submittal by the OU.
3. They find everything to be in place.
4. They approve the sale and it goes live for other OUs.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



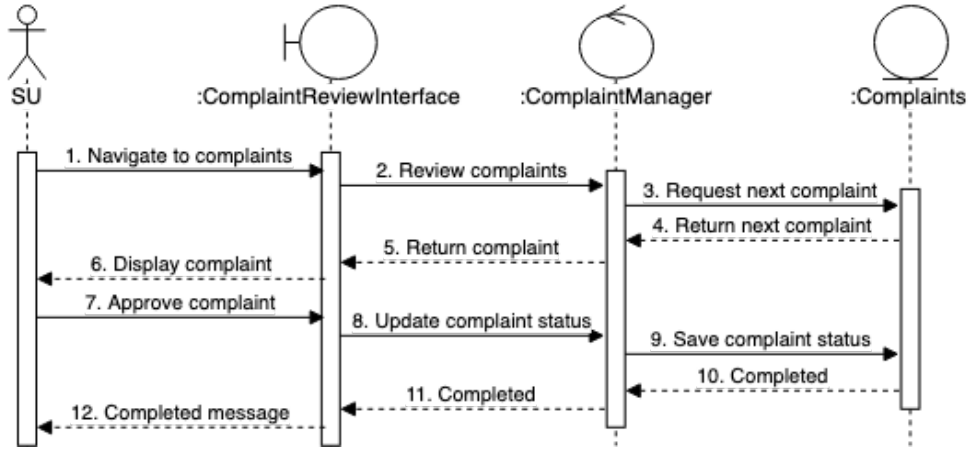
**Use Case 14:** 'Process Complaints'

**Normal Scenario:**

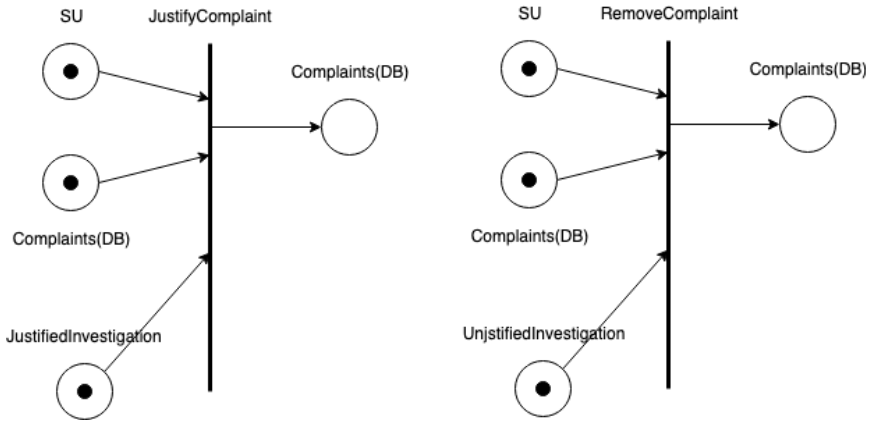
1. An SU logs on to find a complaint has been filed.
2. They navigate to the complaint and read it.
3. It seems a seller didn't send the item on time.
4. The SU reaches out to the seller to find out why the item was delayed.
5. The seller explains that the item weighed more than they initially realized and the shipping time would be increased.
6. The SU judges that the seller should have weighed the item before listing it and marks the complaint justified.
7. The complaint stays associated with the seller's account.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



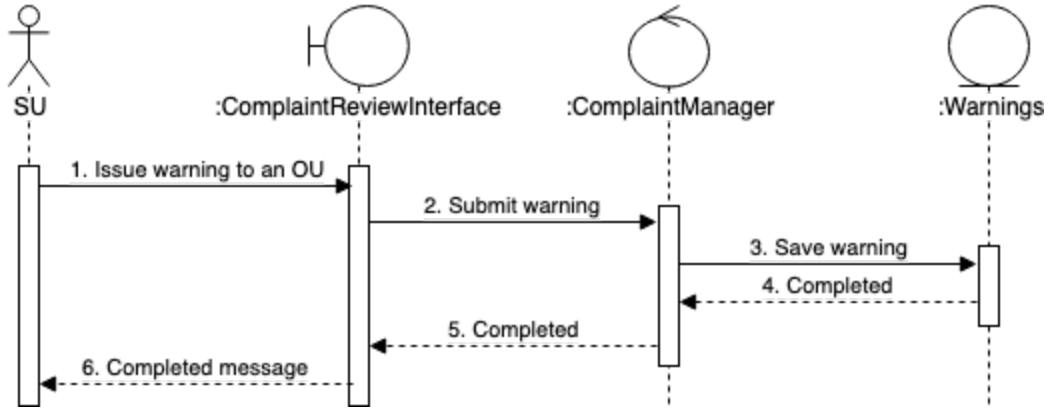
**Use Case 15:** ‘Send Warning’

**Normal Scenario:**

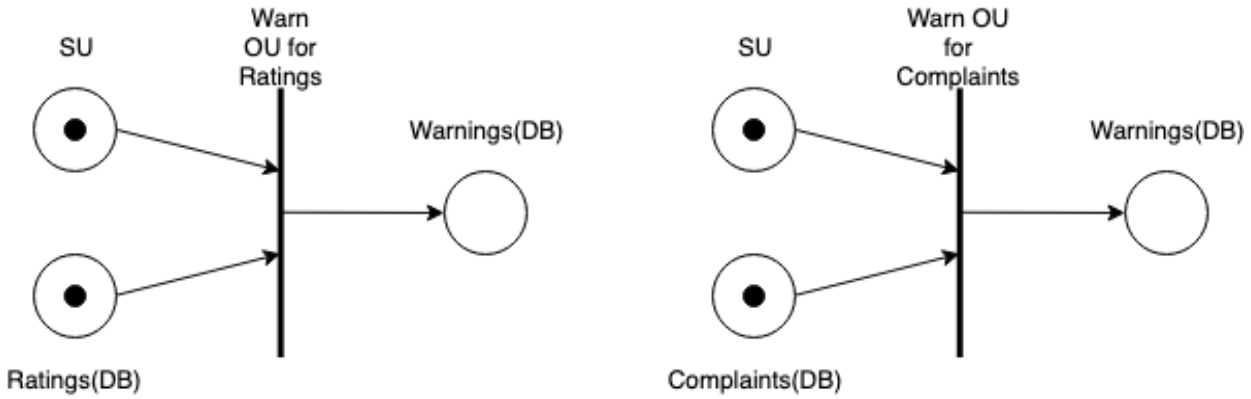
1. After continually sending items late and in bad shape a user’s rating drops below 2 stars.
2. A warning is sent from the SU to the OU.
3. A week later, they receive a second justified complaint associated with their account
4. The account is suspended, but the OU may appeal.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



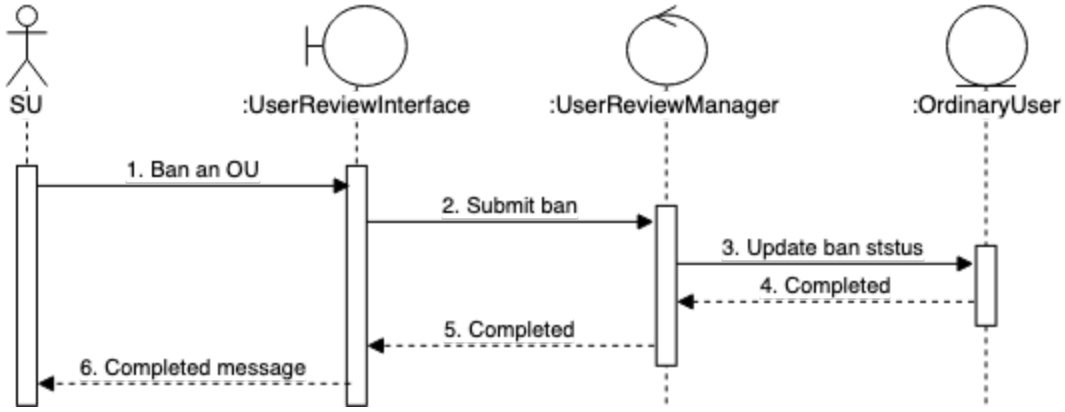
**Use Case 16:** 'Remove User'

**Normal Scenario:**

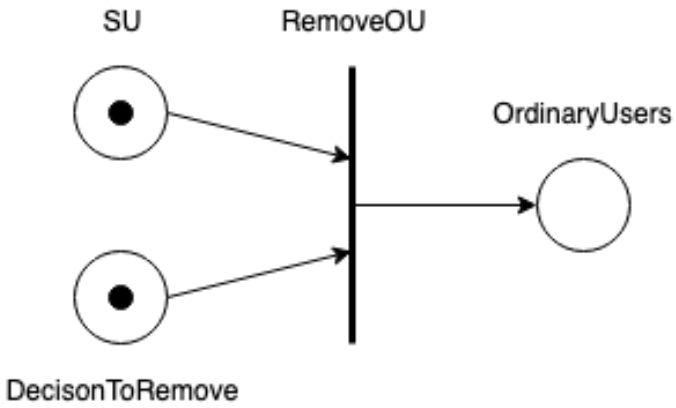
1. After reviewing the appeal of a suspended user the SU decides they should be removed
2. The SU places the OU on a "to-be-Removed" list.
3. Next time the OU logs in, they have only that session to finish up all business.
4. Upon their logging out or closing the application, their account is deleted.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



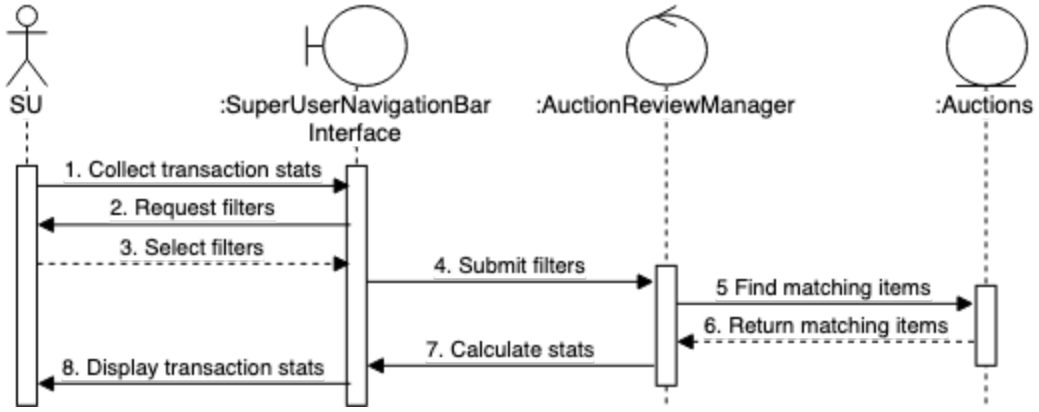
**Use Case 17:** 'Collect Transaction Statistics'

**Normal Scenario:**

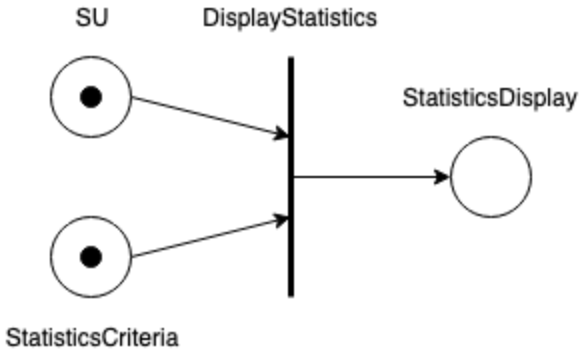
1. The SU needs access to the statistics for the weeks to check the application's success.
2. They maneuver through a few options and filters regarding what statistics to view and for which users.
3. The statistics are clearly displayed in the application.

**Exceptional Scenario:** None.

**Sequence Diagram:**




**Petri-Net:**



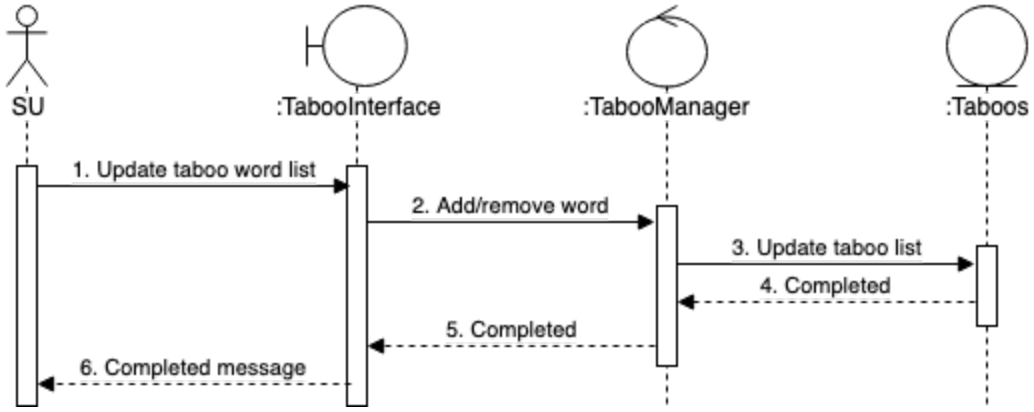
**Use Case 18:** ‘Maintain Taboo Words List’

**Normal Scenario:**

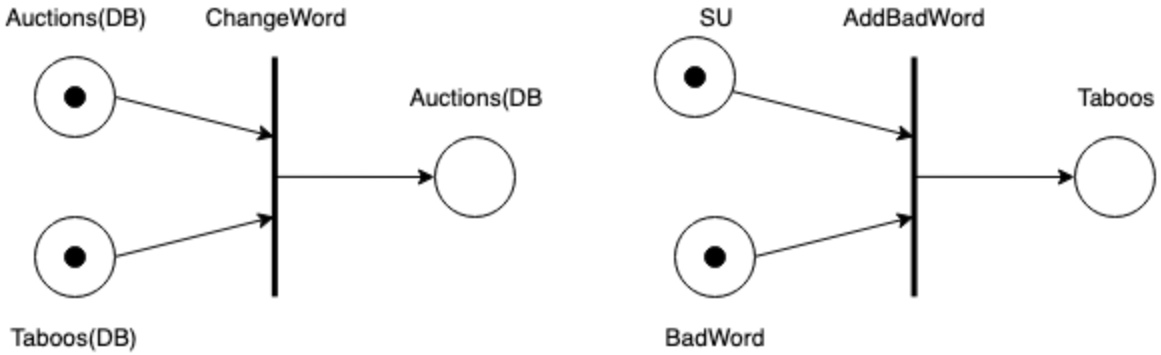
1. The SU updates the word list.
2. Three days later an OU uses the word in an item listing.
3. The autofilter replaces the word in the sentence, and auto issues a warning to the OU.
4. Next time the OU logs they are required to replace the word.

**Exceptional Scenario:** None.

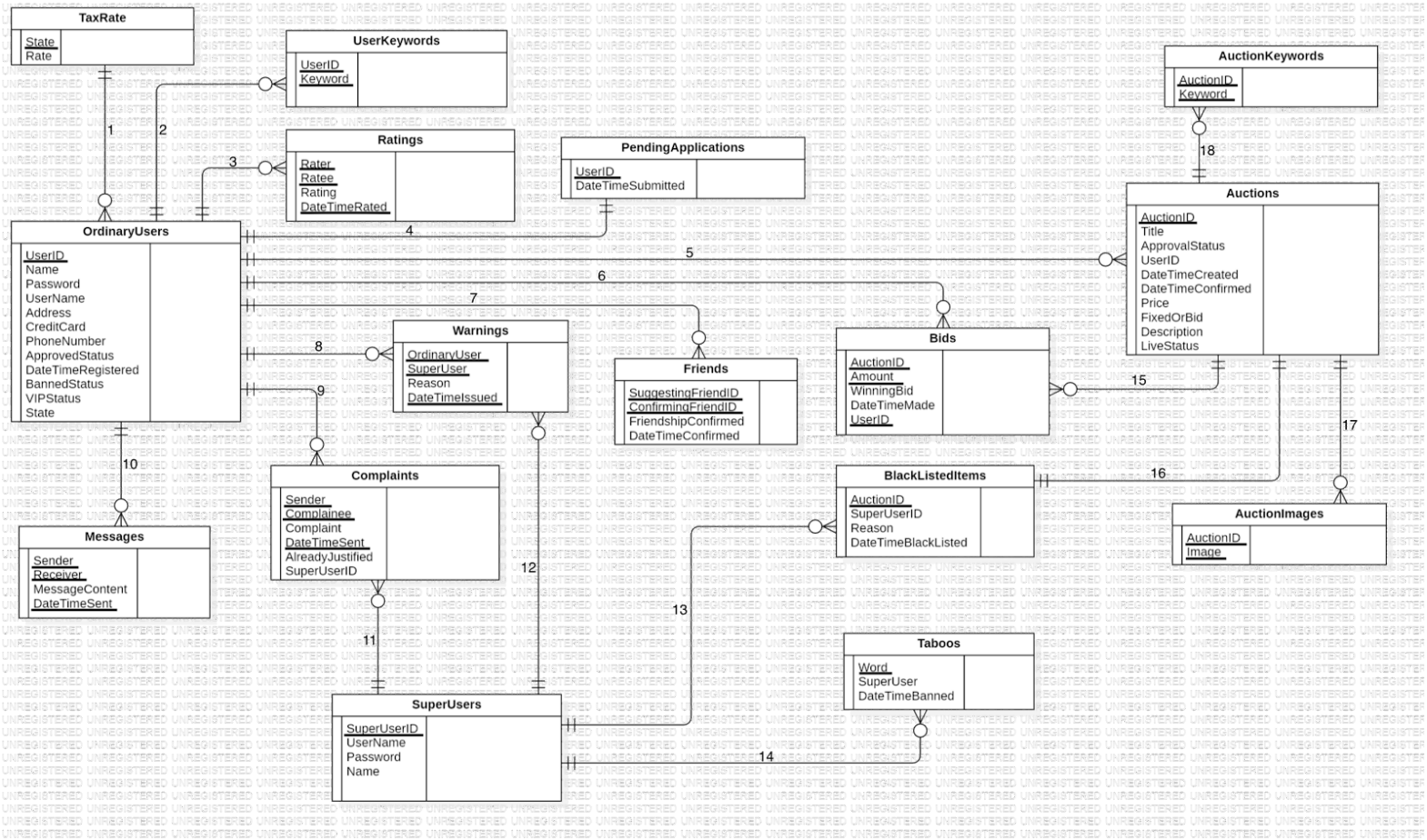
**Sequence Diagram:**




**Petri-Net:**



# 3. E/R Diagram for the Entire System




Each numbered relationship between the entities have the detailed description below:

1. OrdinaryUsers have one TaxRate based on their state
2. OrdinaryUsers maintain UserKeywords
3. OrdinaryUsers have Ratings
4. OrdinaryUsers have PendingApplications
5. OrdinaryUsers can hold Auctions
6. OrdinaryUsers can submit/receive Bids
7. OrdinaryUsers can maintain Friends
8. OrdinaryUsers can receive Warnings
9. OrdinaryUsers can file/receive Complaints
10. OrdinaryUsers can send/receive Messages
11. SuperUsers can receive/process filed Complaints
12. SuperUsers can send Warnings
13. SuperUsers can remove items and put them in BlackListedItems
14. SuperUsers maintain Taboos
15. Auctions can have multiples Bids
16. Auctions could get pushed into BlackListedItems by SuperUsers
17. Auctions should contain at least one AuctionImage
18. Auctions maintain AuctionKeywords

# 4. Detailed Design

As proof of concept, we have created a prototype in Java using the Model–View–Controller (MVC) architectural design pattern. The source code can be found in our git repo (see link at end of this report).

In addition, pseudo code has been provided below for all methods within each class.

**Model Classes:**

Class: User

Intent: Contains the attributes (as well as getters/setters) inherent to all users. Is extended by SuperUser and OrdinaryUser. Contains the methods “LogIn” and “create”

- LogIn(username, password)

*//Username and password are passed to the database. If username and password combo is found in OrdinaryUser or SuperUser table a new OrdinaryUser, or SuperUser object is returned. If no match is found, the function returns the current User object*

- Create(OrdinaryUserObject)

*//pushes the information contained in the ordinary user object (which is constructed by the CreateNewUserManager) into the OrdinaryUser database*

Class: OrdinaryUser

Intent: This is the class representing an ordinary user who is using the system to buy or sell items. They have attributes such as username, password, displayname, creditcard, phonenumber, rating, VIP status, etc.

Many functions of other objects which require the user to be logged in will check this prerequisite by checking that the current User of the system is instanceOf(OrdinaryUser).

Methods:

placeBid(price, auctionObject)

submitAuction(auctionObject)

sendMessage(messageObject)

submitComplaint(complaintObject)

requestFriend(userObject)

acceptFriend(userObject)

updateKeyWord(String keyword)

updateUserInfo(userObject)

gradeUser(rating, userObject)

Pseudocode:

- PlaceBid(price, auctionObject)

  *//Will submit to the bid table the price specified by the user for the given auction*

- SubmitAuction(auctionObject)

  *//will push to the database Auction table the information contained in the auctionObject*

- SendMessage(messageObject)

  *//will push to the database messages table the information contained in the messageObject*

- submitComplaint(complaintObject)

  *//pushes info in complaintObject to the database complaints table*

- requestFriend(userObject)

  *//pushes a new entry into the friends table, where the "accepted bit" is 0*

- acceptFriend(userObject)

  *//searches the friends table for the entry where the requesting friend is the userObject, and changes accepted bit to 1*

- updateKeyWord(string keyword)

  *//adds the specified keyword to the database UserKeywords table for the associated user*

- updateUserInfo(userObject)

  *//creates a new user object with the user specified values, and update the information for the current user in the OrdinaryUser database table with the new information*

- gradeUser(rating, userObject)

  *//adds the rating for the specified userObject to the ratings table*

**Each Model Object not explicitly defined in this document is assumed to have the same attributes described in the attached ER diagram.**

**View Managers:**

Class: DisplayManager

Intent: Constructs and returns the standard view for ordinary and guest users. View is comprised of a JavaFX borderPane with the top component being a HBox with a company Logo, a search bar and submit button for auction searches, and the user's display name.

The left component is the navbar, which is comprised of series of buttons which call the DisplayManager's "changeScene()" function, which in turn control which view is displayed in the boarderPane's center component

The navbar is a view constructed and returned by Class NavBarManager, and the banner is constructed and returned by class BannerManager.

Each view for the center pane is constructed by an associated View Manager.

PsuedoCode:

- DisplayManager(User user){

  *//instantiate the view based on the type of User, instantiating and laying out subplanes*

}

- changeScene(SubScene){

  *//Swchanges the center component of the borderpane to the subscene passed as an argument*

}

Class: NavBarManager

Intent: Constructs and returns the NavBar view for the DisplayManager. Each button in the UI will call the function DisplayManager.changeScene() passing the argument associated with the given button.

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: BannerManager

Intent: Constructs and returns the Banner view for the DisplayManager. This banner is based on the User info which is passed as a parameter to the constructor for BannerManager

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: AuctionSearchManager

Intent: Constructs and returns the view for the auction search results based on search parameters passed to constructor. Each auction has an associated button and text area where the user may submit a bid to the displayed auction. At any time after a bid has been placed, a user may rate the user. It is recommended the user wait until they receive the item before rating the user.

Methods:

- retrieveAuctions(searchParameters)

  *//returns a list of auctions matching the parameters specified,*

- parseAuctions()

  *//parses the list of returned auctions into individual auctionObjects*

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: AuctionCreationManager

Intent: Constructs and returns the view for the user to input the information to be submitted for a auction pending approval. The button to submit the auction will call User.submitAuction() passing the information generated by the “generateAuctionInfo()” function as an argument.

Methods:

- AuctionCreationManager()

  *//instantiate UI*

- GenerateAuctionInfo()

  *//Creates a new auction object based on the information supplied to the fields by the user*

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: MessageManager

Intent: Constructs and returns a view created from the User's inbox. Retrieves all messages from the message table where recipient = username, and populates them in a list. Each message has an associated "reply" button which creates a new messageObject with the user's specified response. Then the button calls User.sendMessage() passing the newly created messageObject as an argument.

Methods:

- createNewMessage(string Username)

  *//in addition to the option specified above vis a vis replying to a message, there is also an option to send a new message by specifying a username to be delivered to before sending*

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: FriendManager

Intent: Constructs and returns a view created using the user's friends. It lists current friends (as queried from the database) as well as provides the option to request a friend, provided the user knows the username of the friend they wish to request. Usernames can be obtained through messaging, auctions, or off-app communication.

- generateUI()

  *//Queries the database for all confirmed friendships where confirming friend OR requesting friend = username, and confirmed = true to populate friends list*
  *//also queries database for all cases where confirmingfriend = username, and confirmed = false in order to populate the list of current pending requests.*
  *//generates a text box and a button to push a new friend request to the table, after the intended friend has been validated as an existing user*

- *getSubScene()*

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: UserComplaintManager

Intent: Constructs and returns a UI for users to submit complaints. They enter a reason into a javaFX “textarea,” and the username of the offending user.

- generateComplaintInfo()

  *//generates a new complaint object when a button is pushed, and calls User.SubmitComplaint() passing the complaint object as an argument.*

- getSubScene()

  *//returns the subscene generated by the class to be displayed by the DisplayManager*

### Class: TransactionHistoryManager

Intent: Constructs and returns a view based on all of the user’s bids and auctions, listed by date, and generated by querying the database.

- getSubScene()

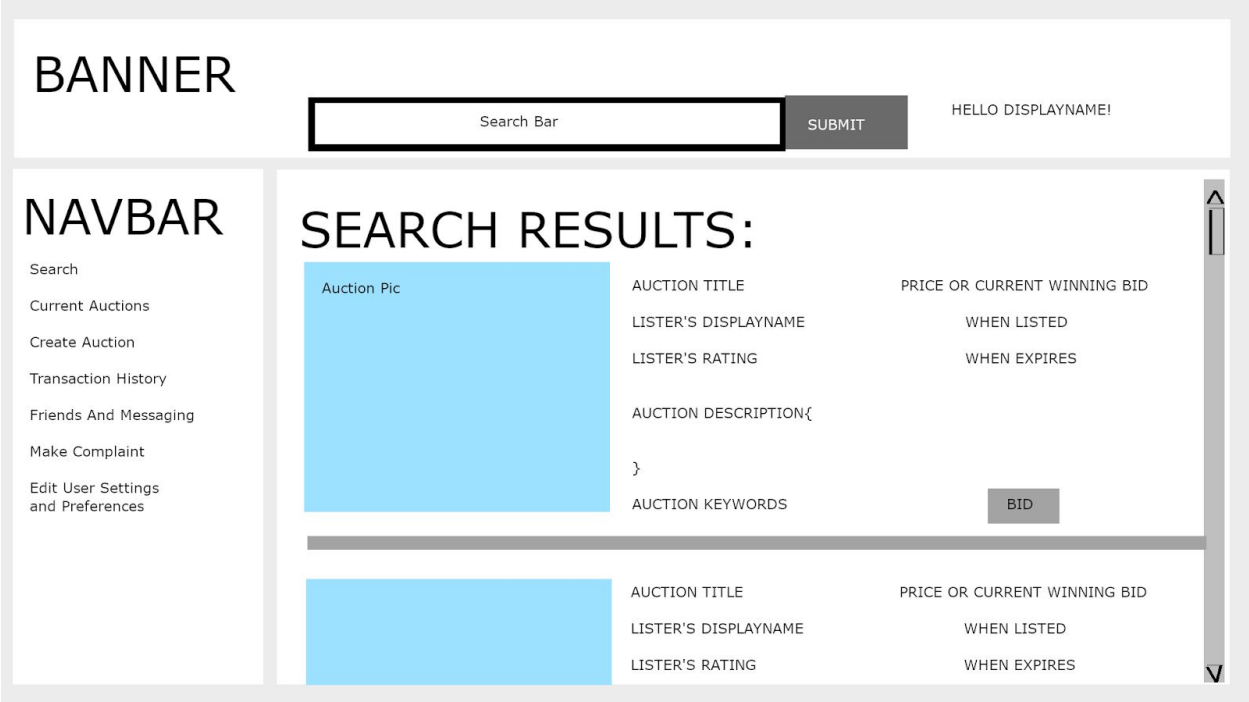
  *//returns the subscene generated by the class to be displayed by the DisplayManager*

# 5. System Screens

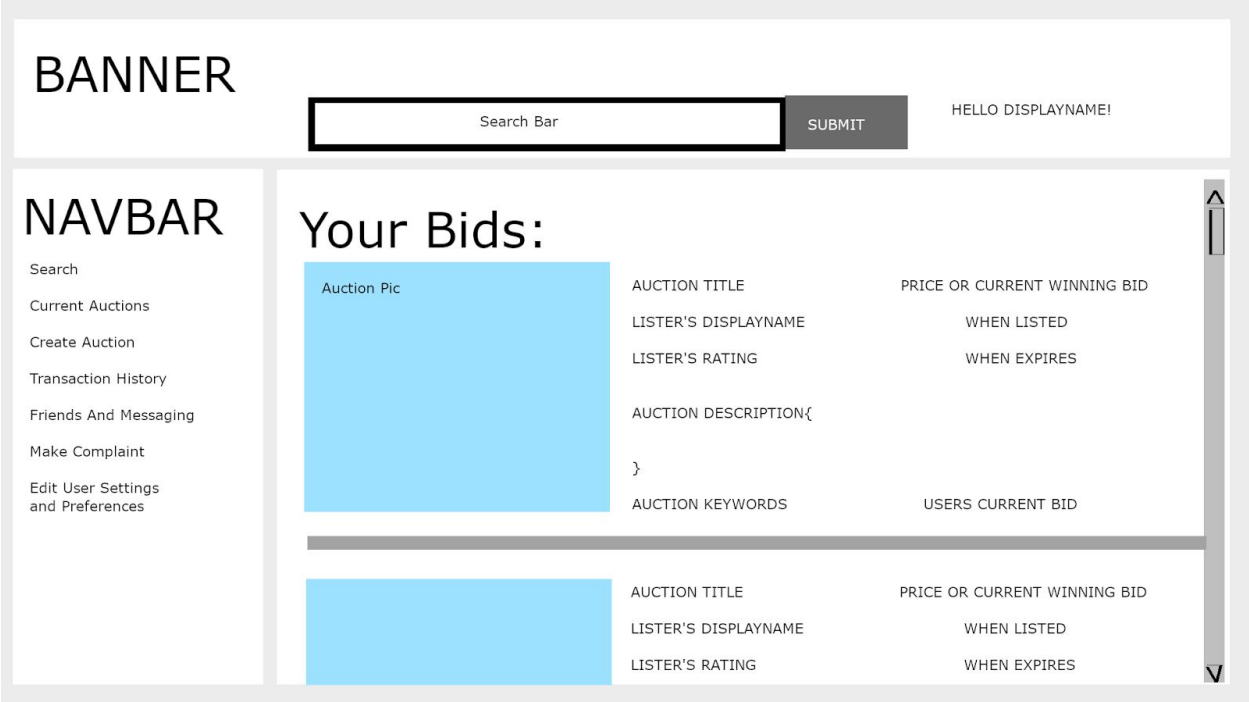
The following mockups illustrate the basic design concept for our GUI interface. A few sample screens are shown here which are to be used as a template for all additional screens.

### UI General Mockup

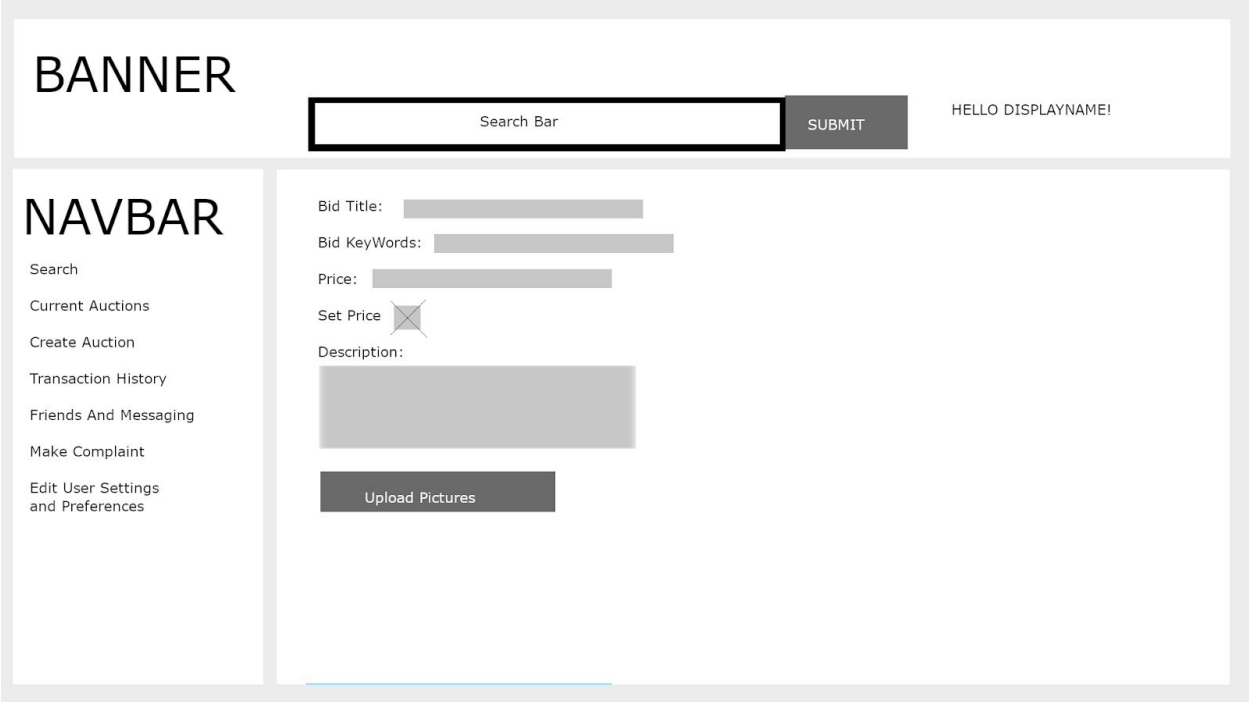
### UI Auction Feed Mockup



### UI User Bids Mockup



### UI Create Auction Mockup



# 6. Minutes of Group Meetings

**April 11, 2019 - Group Meeting 3 (1pm - 4pm)**

1. Went through the Phase II Design Report and discussed how we go about completing the requirements.
2. We decided the first step will be to complete the E/R Diagram as a team.
3. We made our E/R Diagram.
4. Two questions arose while making our E/R Diagram. First, how does one become a super user? How does a SuperUser decide to blacklist an item? We also had a question about the type of E/R diagram and if we need to do it the way instructed in class? Can we do crowfoot?

**April 12, 2019 - Group Meeting 4 (2pm - 4pm)**

1. We decided how our system is going to look.
2. We're going to create a class for each screen view to be displayed and depending on what button you press, a new instance of the scene object will be created.
3. Here are the keys for each entity:
    a. OrdinaryUsers: UserID
    b. SuperUsers: SuperUserID
    c. Messages: Sender + Receiver + DateTimeSent
    d. TaxRate: State
    e. UserKeywords: UserID + Keyword
    f. Ratings: Rater + Ratee + DateTimeRated
    g. Warnings: OrdinaryUser + SuperUser + DateTimeIssued
    h. Complaints: Sender + Complainee + DateTimeSent
    i. PendingApplications: UserID
    j. Friends: SuggestingFriend + Confirming Friend
    k. Auctions: AuctionID
    l. Bids: AuctionID + UserID + Amount
    m. BlackListedIstems: AuctionID
    n. AuctionImages: AuctionID + Images
    o. AuctionKeywords: AuctionID + Keyword
    p. Taboos: Word
4. Next we started on our overall picture of the system using collaboration class diagram.

**April 17, 2019 - Group Meeting 5 (7pm - 10pm)**

1. We started working on the petri-net for each case
2. All the necessary changes/updates were made for each case
3. Also began working on developing the sequence diagrams. Made sure that all cases have petri-net as well as the sequence diagrams
4. There was a discussion of how we would approach the progress of pseudo-code to represent the detailed design
5. Successfully finished designing the E/R Diagram with all of its attributes, keys and relationships
6. Heavily worked on the collaboration class diagram as the overall picture of the system.
7. Some of the GUI screens of the system were done.

**April 18, 2019 - Group Meeting 6 (1pm - 3pm)**

1. Updated our E/R Diagram by adding some attributes
2. Fully completed the design of the collaboration class diagram
3. Continued working on the GUI screens
4. Created a GitHub workspace area for our project
5. Uploaded all of our completed work into GitHub
6. Further discussed of how we would represent the detailed design of the project by writing pseudo-code.
7. Some of the pseudo-codes were developed for methods

**April 19, 2019 - Group Meeting 7 (5pm - 9pm)**

1. The pseudo-codes for the detailed design were all developed
2. Heavily worked and finished on all major GUI screens for the system.
3. Updated the collaboration class diagram
4. Updated some of the petri-nets
5. Some source code was also done as proof of concept
6. Minor issues were addressed and resolved.

# 7. Git Repository

The address of the git repo of our team's work so far:
https://github.com/jaysaduakas/eBamazon