# **Electronic Auction System**

## **1.** **Text Use Cases**

### **Registration**

1. Guest sends request for registration

2. System receives request and responds by redirecting guest to registration form

3. Guest fills the registration form and submits it

4. System checks received data and sends email to Guest

5. Guest receives email and finds security code in that email

6. Using security code Guest validates his/her data

7. If validation passes server confirms account creation

### **Login**

1. Guest sends Login data(username and password) to server

2. System receives data and compares it with data that it has

3. If everything is correct server confirms it and guest become User

4. If something is incorrect server sends error message to the Guest

### **Logout**

1. Guest sends Logout request to server

2. Server confirms it and User becomes Guest

### **Search Item**

1. User/Guest sends search data to server. Search data includes item type and search query

2. Server receives search data and finds all items corresponding to that input data

3. Server constructs the corresponding web page and redirects User/Guest to found item list

### **View Item**

1. User/Guest chooses item from found item list

2. Server receives the request and redirects User/Guest to the item

### **Bid Item**

1. User places his bid for the item

2. System checks if auction is running

3. System accepts bid, if it is greater than the current one

### **Auction Over**

1. System closes auction

2. System chooses User with largest bid and adds item to User’s basket

### **Add Card**

1. User requests to add a credit card to profile and sends request to server

2. Server receives request and redirects User to corresponding page

3. User enters card’s information and submits

4. Server receives that information and confirms

### **Make Payment**

1. User requests to pay for the items in basket and sends request to server

2. Server receives request and redirects User to payment page

3. User chooses card and submits

4. Server makes transaction

## **5.** **System Operation Contracts**

### **Registration**

*registration(personalData)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog is linked with dispatcher

**Postcondition**:

1. creating user:User object with personalData
2. linking user with userCatalog

### **Login**

*login(username, password)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog is linked with dispatcher

**Postcondition**:

1. token, by default null
2. for each user:User in userCatalog
   1. if user.username == username && user.pasword == password then

i. set customer.Status true

ii. generates token

1. return token

### **LogOut**

*logOut(token)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog is linked with dispatcher
3. user:User has Status equal to true and token equal to given token

**PostCondition**:

1. set customer.Status false

### **View Item**

*browseItem(ID)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. store:Store is linked with dispatcher
3. itemType:ItemType is linked with store
4. item:Item with ID is linked with itemType

**PostCondition**:

1. return item

### **Search Item**

*searchItems(typeName, query)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. store:Store is linked with dispatcher

**PostCondition**:

1. result, an empty collection of Item
2. For each itemType:ItemType in Types
   1. if itemType.name == typeName

i. For each item:Item in ItemList

i. If item description or item name contain query, push item into

result

1. return result

### **Bid Item**

*bidItem(token, ID, price)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog linked with dispatcher
3. user:User linked with userCatalog, having token
4. auctionList:AuctionList linked with dispatcher
5. auction:Auction linked with auctionList and user and item
6. price is greater then auction.price and item.price
7. user.canParticipate must be true

**PostCondition**:

1. auction.user = user
2. auction.price = price

### **Auction Over**

*auctionOver(price)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog linked with dispatcher
3. user:User linked with userCatalog, having token
4. auctionList:AuctionList linked with dispatcher
5. auction:Auction linked with auctionList and user and item

**PostCondition**:

1. user.canParticipate = false

### **Add card**

*addCard(cardInfo,token)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog is linked with shop

**PostCondition**:

1. For each user:User in userCatalog
   1. If user.status == true and user.token == token
      1. create card:Card and add it to user

### **Make Payment**

*makePayment(token, ID)*

**Precondition**:

1. dispatcher:Dispatcher is underway
2. userCatalog:UserCatalog linked with dispatcher
3. user:User linked with userCatalog, having token
4. auctionList:AuctionList linked with dispatcher
5. auction:Auction linked with auctionList and user
6. archive:Archive linked with dispatcher

**PostCondition**:

1. archiveRecord:ArchiveRecords is created
2. archiveRecord is initialized with corresponding data
3. auction is destroyed
4. user.canParticipate set to true