**Final Report**

Team Buyhigh

Youngchan Lim

Jaewon Lee

1. **General Description**

* Goal: Creating a simple Auction application where users can deal products with each other by bidding a price. Users may upload products, bid for a product, and finally make a final order for the product.
  1. Application Domain
     1. OS: Windows
     2. Type: Web application
  2. Required Technologies:
     1. IDE: Visual Studio Code
     2. Front-end: React
     3. Back-end: Node.js and express
     4. Server: MYSQL

1. **Use Cases**
   1. Register

|  |  |
| --- | --- |
| **Actor** | User |
| **Goal** | User creates an account with hashed password and store to database |
| **Precondition** | User is not registered to the site and wish to sign up |
| **Trigger** | Click on Register button on the header bar |
| **Scenario** | 1. User enters required information 2. User clicks ‘Register’ button 3. Application redirects the user to main screen |

1. Login

|  |  |
| --- | --- |
| **Actor** | User |
| **Goal** | User logs in and be able to use the application |
| **Precondition** | User not logged in |
| **Trigger** | User enters the site |
| **Scenario** | 1. User gets redirected to login screen 2. User enters username and password 3. User clicks LOGIN button 4. Application redirects the user to main screen |

1. Logout

|  |  |
| --- | --- |
| **Actor** | User |
| **Goal** | User logs out and return to the initial state |
| **Precondition** | User logged in |
| **Trigger** | User clicks on Logout button on header bar |
| **Scenario** | 1. Application redirects the user to login screen |

1. Profile

|  |  |
| --- | --- |
| **Actor** | User |
| **Goal** | User can view and change own information stored |
| **Precondition** | User logged in |
| **Trigger** | User clicks Profile button on header bar |
| **Scenario** | 1. User types in new information 2. User clicks Update button at the bottom 3. User information is updated |

1. Home

|  |  |
| --- | --- |
| **Actor** | User |
| **Goal** | User can see list of opened auctions |
| **Precondition** | User is logged in |
| **Trigger** | User clicks HOME button on menu bar |
| **Scenario** | 1. User navigates through home screen |

(Auction)

1. Open Auction

|  |  |
| --- | --- |
| **Actor** | Seller |
| **Goal** | User can open auction to display |
| **Precondition** | User logged in |
| **Trigger** | User clicks Open Auction button on header bar |
| **Scenario** | 1. User fill out auction information 2. User clicks OpenAuction button at the bottom of the form 3. Auction is listed on the homepage |

1. Bid

|  |  |
| --- | --- |
| **Actor** | Buyer |
| **Goal** | User bids to opened auctions |
| **Precondition** | User logged in |
| **Trigger** | User clicks on auction box in Home |
| **Scenario** | 1. User checks Current Bid Price 2. User checks Auction End Date 3. User bids |

1. Offer

|  |  |
| --- | --- |
| **Actor** | Buyer |
| **Goal** | User offers to opened auctions |
| **Precondition** | User logged in |
| **Trigger** | User clicks on auction box in Home |
| **Scenario** | 1. User checks Current Bid Price 2. User checks Auction End Date 3. User bids |

1. Bid History

|  |  |
| --- | --- |
| **Primary Actor** | Buyer |
| **Goal** | User views current max-bid and user’s max-bid of item |
| **Precondition** | User is logged in and bid in auction |
| **Trigger** | User clicks MyBids button in header bar |
| **Scenario** | 1. User bids in auction 2. MY BID is updated to the following amount |

1. Offer History

|  |  |
| --- | --- |
| **Primary Actor** | Buyer |
| **Goal** | User views current max-bid and user’s offered price of item |
| **Precondition** | User is logged in and offered in auction |
| **Trigger** | User clicks MyBids button in header bar |
| **Scenario** | 1. User offers a price in auction 2. MY OFFER is updated to the following amount |

1. Sell History

|  |  |
| --- | --- |
| **Primary Actor** | Seller |
| **Goal** | User views current max-bid and offered price of item uploaded |
| **Precondition** | User is logged in and opened auctions |
| **Trigger** | User clicks MySells button in header bar |
| **Scenario** | 1. User offers a price in auction 2. MY OFFER is updated to the following amount |

1. Accept Offer

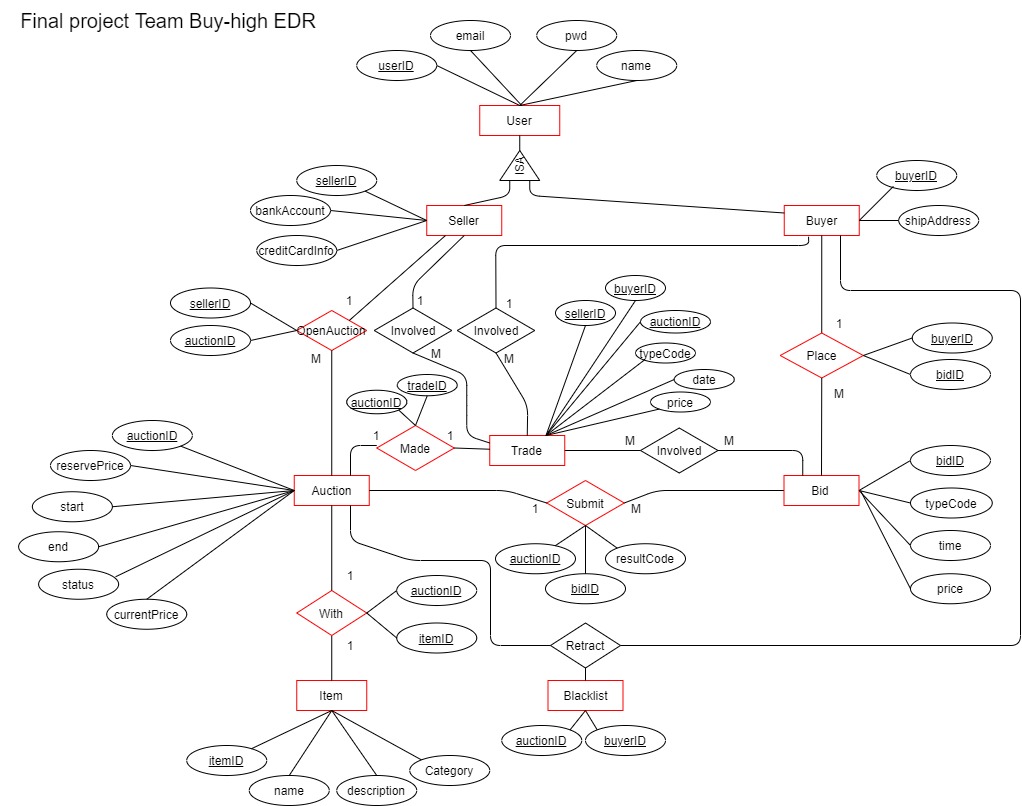
|  |  |
| --- | --- |
| **Primary Actor** | Seller |
| **Goal** | User accepts to the offered price and makes a deal |
| **Precondition** | User logged in and offered a price for an auction |
| **Trigger** | User clicks Accept button in MySells page |
| **Scenario** | 1. User accepts the offer and deal is made 2. Trade information is sent to both buyer and seller 3. Following auction closes 4. User is redirected to trades page |

1. Trades History

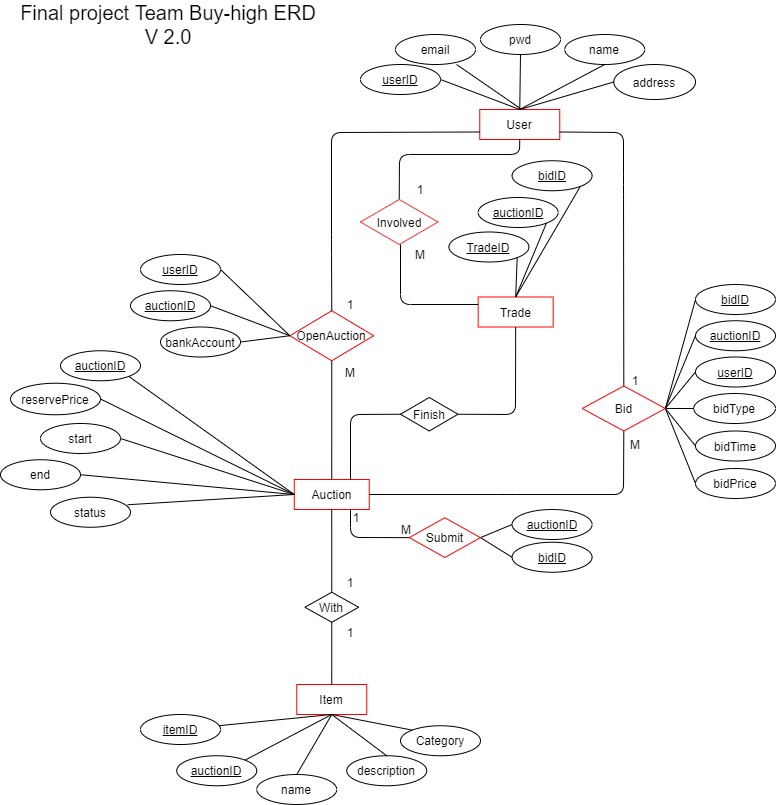
|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Goal** | User views participated auctions that have been closed |
| **Precondition** | User logged in and participated in auctions and the auctions are closed |
| **Trigger** | User clicks MyTrades button in header bar |
| **Scenario** | 1. Buyer checks the list of bought items 2. Buyer sends amount in BILL to corresponding DEPOSIT ACCOUNT 3. Seller checks the list of sold items 4. Seller sends the item to the corresponding SENDING ADDRESS |

1. Auctions Close

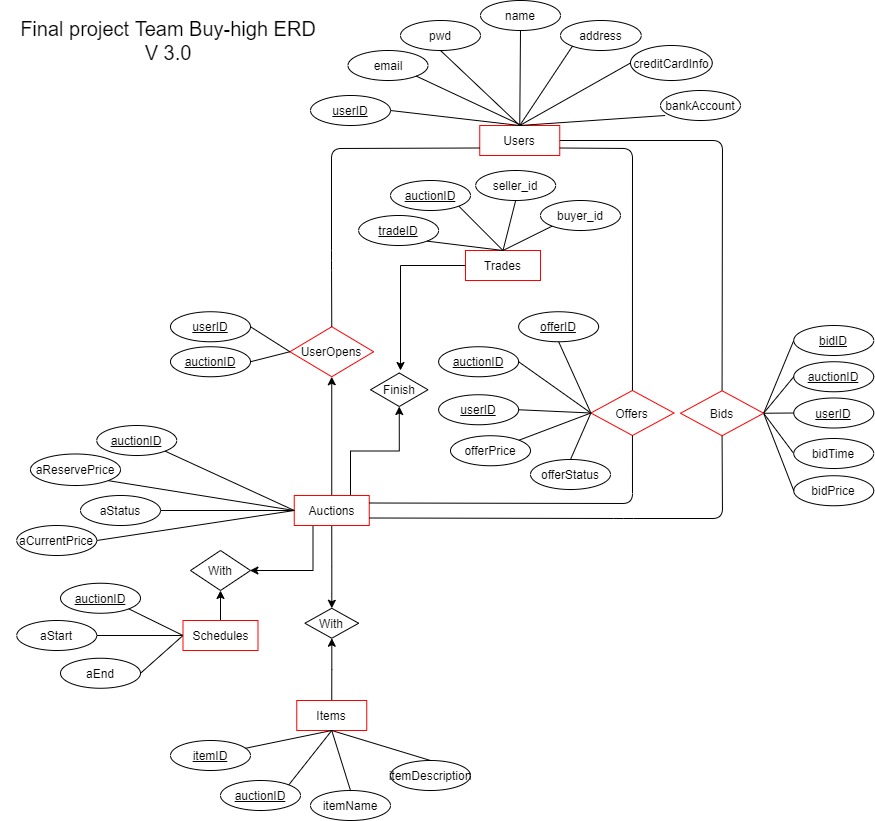
|  |  |
| --- | --- |
| **Primary Actor** | Seller |
| **Goal** | Auctions is closed and removed from Home |
| **Precondition** | Auction have been opened |
| **Trigger** | 3days pass after the Auction open |
| **Scenario** | 1. Auctions automatically closes after 3days of opening 2. The deal is made to the highest bidder 3. Information is stored in MyTrades 4. Information is removed from MyBids and MySells |

1. **E-R Diagram** ****

ERD version 1



ERD version 2



ERD Final version 3

1. **Database Normalization**

- ER-Diagram from *Progress Report 1* and *Progress Report 2* and this document shows three versions of ERD that were modified in progress.

- We eliminated all the involved actions in ERD version 1

- User table was separated to Seller and Buyer using ISA but we figured that there were many redundancy using userId, sellerId, and buyerId therefore we reduced redundancy in relationships with other tables (like Place table in ERD version 1).

- In User table, userId is primary key and {username, password} works as a determinant, however; for functional use, we decided that it was inevitable to keep it as it is.

- When we first designed out tables and ERD, we noticed many redundancies in many tables especially in Auction table and complication in relationship in bid, trade, made, etc. Our first design of Auction table is shown below.

|  |
| --- |
| **Auction** |
| aid(pk): A, name: N, description: D, category: C, status: S, currentPrice: P, reservePrice: R, startDate: T, endDate: E |
| FD: A->NDCSPRTE, N -> DC, T->E |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| auctionId | name | description | category | status | currentPrice | reservePrice | startDate | endDate |
| 1 | Cup | Good cup | Home | Open | 100 | 1000 | … | … |
| 2 | Desk | Hard Desk | Home | Open | 500 | 2000 | … | … |
| 3 | Book | Nice book | Book | Close | 10 | 10 | … | … |
| 4 | Note | Neat Note | Book | Open | 5 | 20 | … | … |

[Fig. 1] Example of Auction Table

1. 1st Normal Form
   * Each cell is single valued
   * Entries in a column are same type
   * Row uniquely identified by auctionID
2. 2nd Normal Form
   * All attributes do depend on the key however, we figured that it might be redundant to have both auctionId and name in the same table since user cannot duplicate same item many times. Therefore, we decided to decompose AUCTION table into two tables like below.

Auction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| auctionId(pk) | Status | currentPrice | reservePrice | startDate | endDate |
| 1 | Open | 100 | 1000 | … | … |
| 2 | Open | 500 | 2000 | … | … |
| 3 | Close | 10 | 10 | … | … |
| 4 | Open | 5 | 20 | … | … |

Item

|  |  |  |  |
| --- | --- | --- | --- |
| itemId(pk) | Name | Description | Category |
| 1 | Cup | Good cup | Home |
| 2 | Desk | Hard Desk | Home |
| 3 | Book | Nice Book | Book |
| 4 | Note | Neat Note | Book |

Relation Table

|  |  |
| --- | --- |
| auctionId | itemId |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

1. 3rd Normal Form
   * All fields should be determined Only by Key in the table but we figured out that startDate always determines the endDate. So we decomposed AUCTION table and made another table called SCHEDULE.

Auction

|  |  |  |  |
| --- | --- | --- | --- |
| auctionId(pk) | Status | currentPrice | reservePrice |
| 1 | Open | 100 | 1000 |
| 2 | Open | 500 | 2000 |
| 3 | Close | 10 | 10 |
| 4 | Open | 5 | 20 |

Schedule

|  |  |  |
| --- | --- | --- |
| startTime(pk) | endTime | auctionId(fk) |
| … | … | 1 |
| … | … | 2 |
| … | … | 3 |

* + For Trades table,

|  |
| --- |
| **Trades** |
| Aid(fk): A, sellerId(fk): S, buyerId(fk), B, tradeId(pk): T, Date: D, shipAddress: H, bankAccount: K, price: P |
| FD: T -> ASBDHKP, B->H, S->K, A->DP |

* + To remove unnecessary functional dependencies, we simply removed redundancies for H, K, and DP.
  + We removed relation with Seller and Buyer and simply merged with Bid table so we can get all the user information through Bid table.

To summarize,

* + Auction table decomposed to Auction, Item, and Schedule tables
  + Redundant attributes in Trades were deleted and simplified
  + All other tables were designed to have unique determinants as superkey therefore are in BCNF form.
  + Therefore, all tables are normalized.

1. **User’s Manual**
   1. Download the latest version of Xampp from <https://www.apachefriends.org/index.html>.
   2. Open XAMPP and start Apache and MYSQL Module.
   3. Click shell and type in mysql -u root -p and password of your own. Password will be

* No password. Just press enter
* Type in root.
* One that you have set.
  1. Type in MariaDB [(none)]> create database auction;
  2. Download both server and client folder.
  3. Open terminal(cmd) from your device and go to the directory of the folder.
* First, go to server folder by “cd (folder location)/server”
* Then type in “npm install”.
* Then type in “npm install jsonwebtoken”.
* Then type in “npm start”.
* Then open up new terminal window without closing the previous one and go to client folder as you did for server folder.
* Type in “npm install”.
* Then type in “npm start”
  1. If successfully done, you will see a web-page opened with your default browser. Register and Log-in to use the functions.

1. **Goal that were made and were not**
   1. Our original goals were made:
      1. User management (All made)

* Registration: User will be able to create an account to use the website
* Log-in: User will be able to log in to the system by creating an account
* User-info: Users will be able to view user profile which contains user’s data including name, id, address, and pay information
* User-info update: Users will be able to edit the information on the User-info page
  + 1. Product management
* Create auction: Users will be able to upload products to sell. Users will have to set the start price, name, and description of the product.
  + 1. Auction management
* Check Auction status: This shows the status of the auction such as current bid, product detail, bidding history.
* Place bid: Placing bids have a least-price restriction which is 5% of the current bid. In other words, the buyer should place a bid at least higher than 5% of the current one.
* Direct buy(Offer): The buyer could offer a direct buy price to the seller secretly, however the seller could only receive the highest offer.
  + 1. Order management
* Order: When the deal is made, the ordering information is stacked to the system.
  1. Goals not made and Possible improvement:
     1. Adding category attribute to the item table
* For improvement: If time allows, we will implement a sorted list page where user can view items according to selected category.

1. Code complexity in some of routes functions

* For improvement: There are many if statements and for loops nested to each other which might cause run time issue when lot of users access the server. Therefore, we will try to simplify these statements.

1. Retraction function

* For improvement: We originally planned to put a retraction function for the bidding but we gave other functions priority to implement due to functional significance. We will implement this function later for completeness.

1. UI

* Not enough time to design and fit everything neatly. For completeness, will make it more professional.

1. Additional features
   1. Deletion of ongoing Auction
   2. Deletion of User
   3. Admin functions to manage users and items
   4. Make available to upload photos of items
   5. Inter-User Chatting functions for communication of buyer and seller
2. **Code will be Handded out with this document. Open server and client folder to see the codes.**