Viraj Dongaonkar
Professor Nandakumar
EE422C- Software Design & Implementation II
November 27, 2020

Final Project Documentation

Programmer's Point of View:

- **Mode of use:**
    - This program can be run through the executable JAR files. Extra files such as item images and sound files should be placed in the same directory as the JAR file The database is created using SQL in the Server directory whenever Server is run. There is a file called "input" where all of the items to be initialized will be read in by the Server. If one is to change the items, this is the file in which to do so.
- **Required Features:**
    - Server handles multiple customers and multiple items. There is no upper bound strictly defined, so there can be as many customers and items as the user desires.
    - Communication occurs purely through sockets where each Client has a thread that looks for incoming communication from the Server and the server communicates information to all sockets when a change is made.
    - Server reads in from file and inputs information into the SQL database at runtime
    - If the Buy It Now price is reached or time has run out for an item, it will be clearly displayed that the item is no longer up for bid. If a Client attempts to bid on it still, they are reminded of the reason why their bid did not go through.
    - Valid bids are accepted and the Server notifies all Clients when a valid bid occurs.
    - JavaFX GUI utilized for Client class.
    - The termination mechanism for items is either if the timer runs out on the item or the buy it now price has been reached.
    - All Clients are notified of who bought the item when the program terminates. If no one has bought it, that will be the item's notification status.
    - There is a Quit button that kills all threads for that Client instance whenever it is pressed. This allows for a smooth termination of the specified GUI and its threads.

- Synchronization is employed in order to maintain thread safety between the Clients when updating the database or sending messages to clients.
- No Exceptions are thrown under normal circumstances.
- All class variables are private and have getters and setters. Variables are named appropriately and in camelback text. These are just a few of the ways that good style and OOP principles are adhered to in this project. Synchronization also is adhered to in order to ensure there are no issues when multithreading.

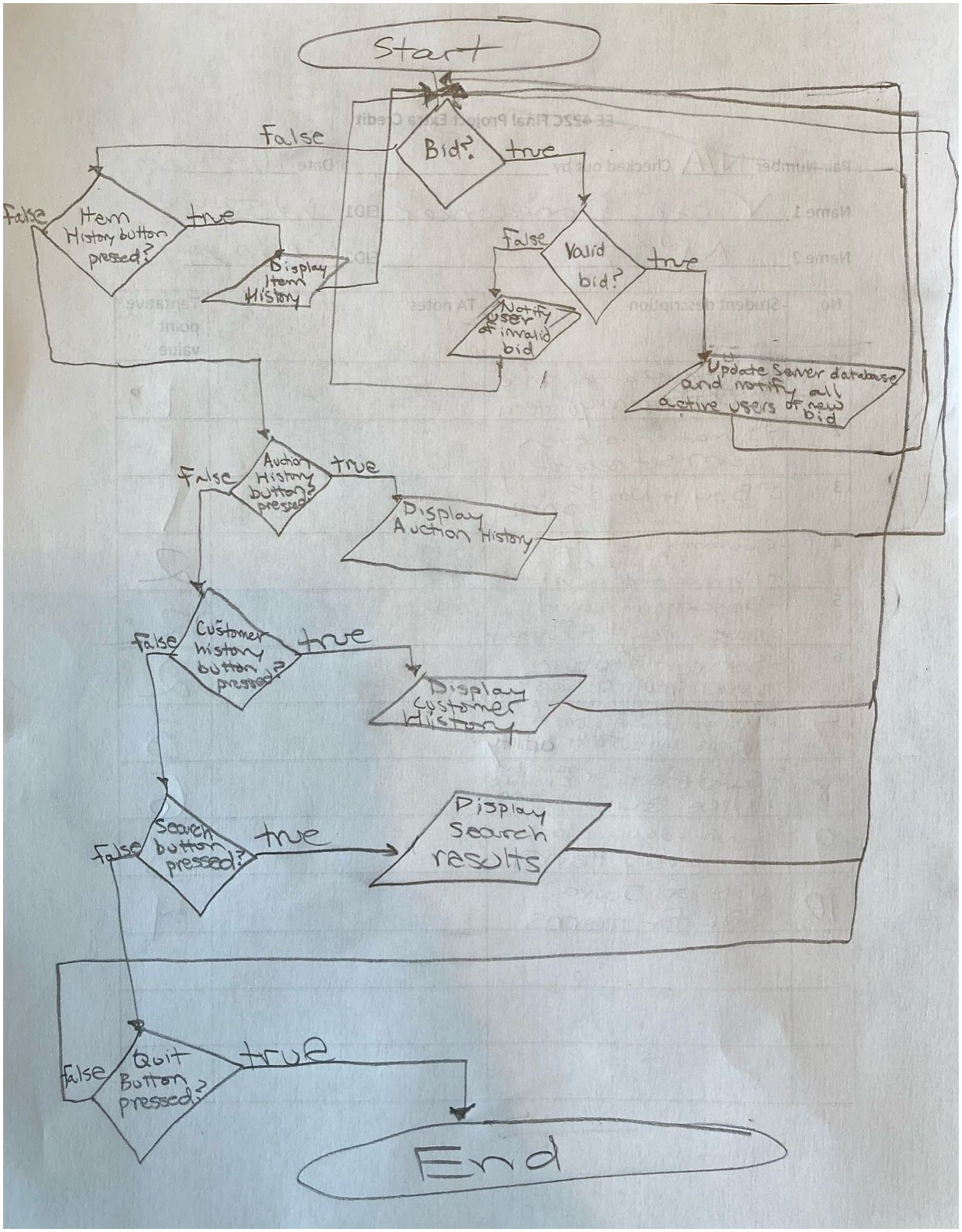
- **Optional Features:**
  - Minimum nonzero price is read in from the input file as an additional parameter.
  - Duration of each item is read in from the input file as an additional parameter.
  - "Buy It Now" price is read in from the input file as an additional parameter.
  - Every Client can see the history of bids on every item. This is sent as a request from the Client to the Server. After that, the Server sends the packaged information back to the Client, and that information shows up on the GUI. The Client will also be able to see if the item has been sold yet.
  - Descriptions and images associated with each item are taken by the Server from the input file, and are sent to the Client in order to have these features show up on the GUI.
  - There is a search feature that checks to see if there is an item and/or customer that matches the query. The query is sent from the Client to the Server and the Server sends back the answer to the question.
  - There is a nonvolatile auction history and customer activity log that can be shown on the Client GUI if the associated buttons are pressed.
  - Sound effects are employed when a bid is successful or when an item is no longer able to be bid on.
  - Each item has an individual countdown clock with different values that is continuously updated on the GUI by having the server update this value every second and send it to the Client.
  - Java SQL is used in order to store the data on the items. This data is sent to a Client whenever the client logs in. It is also continuously updated so that the new Client will receive up-to-date information.

User's Point of View:

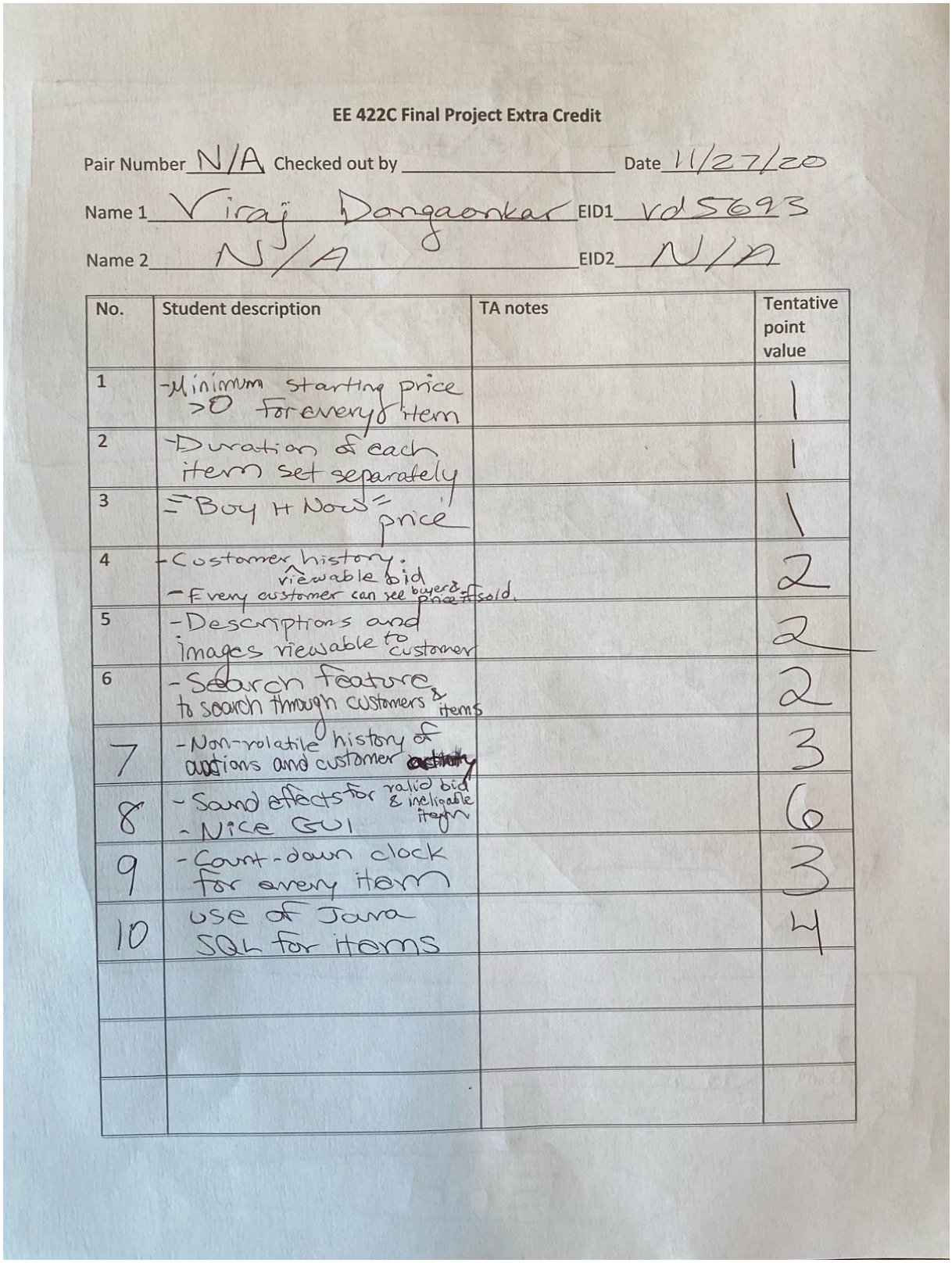
- **Mode of use:**
    - This program can be used by first clicking the Server JAR file and then clicking the Client JAR file. A database is created after running the Server file, but the user does not need to worry about that aspect of this program. If there are any additional images associated with the items the user is bidding on, they should place the images in the same directory as the Client JAR, and update the "input" file that is in the Server JAR directory with the name of the corresponding image.
- **Required Features:**
    - After the server is set up, the user can start the Client program and log in with a username
    - Whenever one customer makes a bid, the Server stores the update and notifies all customers
    - Server reads from input file to set up initial information for items
    - There is a Buy It Now price for every item that a user can pay in order to secure the item.
    - The customer will be notified when a valid bid goes through
    - Items can no longer be bid on if their timer runs out or if the Buy It Now price has been reached
    - The customer will be informed of who bought an item after the item cannot be bid on anymore. If no one purchased it, the customer will be notified of that as well.
    - There is a Quit button that allows the customer to end their bidding session.
    - Bids will be processed properly and in an orderly fashion.
- **Optional Features:**
    - Initial minimum prices for each item that come from the input file are shown to the first customer at startup.
    - Duration of each item that comes from the input file is shown to all customers.
    - "Buy It Now" prices for all items that come from the input file are shown to the customers.
    - Every customer can see the bid history of a certain item. If someone bought the item, the buyer and selling price are shown to every customer.

- Descriptions and images associated with each item that come from the input file are shown to all customers.
- Customers can search for certain items and/or customers when they press the Search button.
- There is a nonvolatile auction history and customer activity log that the customer can see if they click on the buttons that request this information.
- The customer hears sound effects when a bid is successful or when an item is no longer able to be bid on.
- Each item has a different time limit for how long it is valid to bid on
- Every customer's valid bids update the database of items for the server.

Diagram Showing Flow Based off User Decisions:



Extra Credit Sheet:



EE 422C Final Project Extra Credit

Pair Number N/A Checked out by _____ Date 11/27/20

Name 1 Viraj Dangaonkar EID1 vd5693

Name 2 N/A EID2 N/A

No.	Student description	TA notes	Tentative point value
1	-Minimum starting price >0 for every item		1
2	-Duration of each item set separately		1
3	-"Buy It Now" price		1
4	-Customer history: viewable bid -Every customer can see buyer & price it sold.		2
5	-Descriptions and images viewable to customer		2
6	-Search feature to search through customers & items		2
7	-Non-volatile history of auctions and customer activity		3
8	-Sound effects for valid bid & ineligible item -Nice GUI		6
9	-Count-down clock for every item		3
10	use of Java SQL for items		4

References

https://stackoverflow.com/questions/47183795/how-does-platform-runlater-functions

https://stackoverflow.com/questions/4044726/how-to-set-a-timer-in-java

https://stackoverflow.com/questions/26305/how-can-i-play-sound-in-java

https://www.tutorialspoint.com/javafx/javafx_images.htm