The project was built using Solidity which is an object-oriented programming language used to write smart contracts.[5] The project was built in Remix which is the IDE of a blockchain-based platform called Ethereum.[6][7]

Open-sourced resources were imported into the IDE to create the base code of blockchain-based auction that was used to set a foundation in order to add or remove features of the auction.[8] The open-sourced code was run to see the basic functions of the auction and it was noticed that an address or account was not being given to the seller. However, the reason for this was just human error. The rest of the code ran without any difficulty, however, in order to implement the Vickrey auction component, some changes were made.

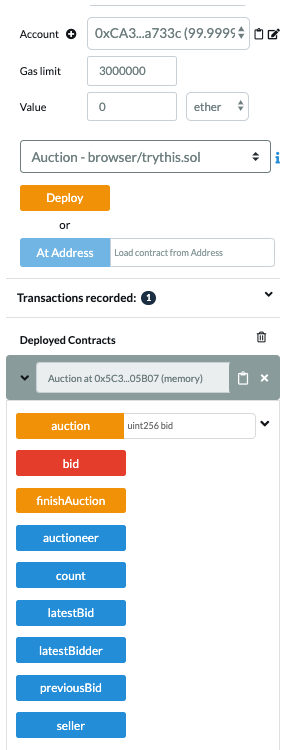
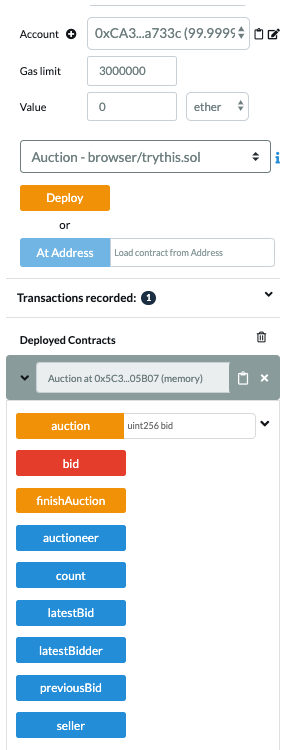
A new variable named “previousBid” was created and it was set to another variable, “latestBid”, which was equal to the most frequent bid. The creation of this variable allowed for a place to store the previous bid which would be necessary when the last bidder has to pay the auction of the previous bid. This variable was placed right before the value of the new bid was set equal to “latestBid”.

|  |
| --- |
| previousBid = latestBid;  latestBid = msg.value; |

**Figure 2:** This is what the code to set up “previousBid” looked like.

Another variable named “count” was also added; its purpose was to keep track of the number of bids placed. Determining the count was important because the addition of the code from above allowed a sole bidder to not pay any ethers, which are the currency of Ethereum. When the auctioneer opened an auction, the count was set to zero and whenever a bid was placed, the count increased by one. So if the count was equal to one, then the sole bidder paid what they bid, but if there were more than one bidder, then the last bidder paid the second-highest bid.

|  |
| --- |
| constructor() public {  auctioneer = msg.sender;  count = 0;  }  .  .  .  previousBid = latestBid;  latestBid = msg.value;  latestBidder = msg.sender;  count = count + 1; |

**Figure 3:** This is what the code to set up the count looked like. It was also used to help determine the amount of ethers a bidder paid. 

**Figure 4:** This is the console that helped run the auction. Each blue “button” in the console were the variables declared in the code. If those buttons were pressed, then the address or value currently assigned to that respective variable could be seen. The orange “auction” button assigned the seller and the minimum bid, and the orange “finishAuction” button ended the auction. The red “bid” button sets a bid.

There are already five addresses or accounts preset in the IDE, each with 100 ethers to spend. An address was selected from the account dropdown to open the auction, and then the “deploy” button is pressed. The selected address becomes the auctioneer and the auctioneer was the only address that was also able to close the auction.

After the auction was opened, another address was selected to become the seller, a minimum bid value was entered into the space beside the “auction” button, and the button was clicked to set the seller and the minimum bid. The rest of the addresses were the bidders. To place a bid, an address had to be selected and a value, which was greater than the value of the previous bid, was entered into the value box and the red “bid” button had to be clicked; this cycle of bidding continued until the bidders would not go further. Then as mentioned above, the auctioneer was selected to close the auction by pressing the orange “finishAuction” button.