APPENDIX LIST

Appendix 1 – Curriculum Vitae

Personal Data

Name : Ningsih Adityas

Place, Date of Birth : Bekasi, 20 March 2000

Gender : Female

Religion : Islam

Citizenship : Indonesian

Address : Jalan Melon Barat Blok DD7 No.3, Bekasi Utara 17121

Phone Number : 0812-8100-0220

E-mail : Ningsihadityas15@gmail.com

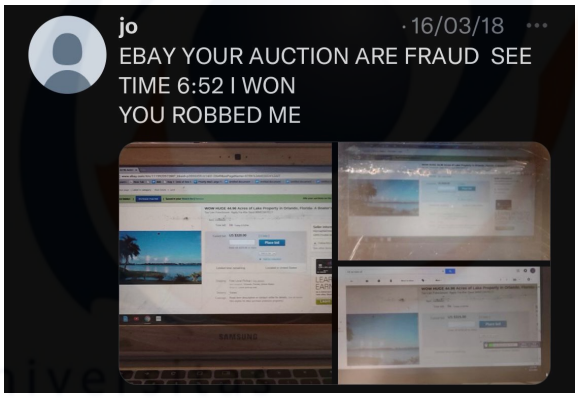
Education History

Year			School/Institution	Major	Education Level
2006	-	2009	SDN Teluk Pucung 11 Bekasi	-	Elementary School
2009	-	2012	SDN 04 Jakarta	-	Elementary School
2012	-	2013	SMPN 2 Jakarta SSN	-	Junior High School
2013	-	2015	SMP Mutiara 17 Agustus 1 Bekasi	-	Junior High School
2015	-	2018	SMA Mutiara 17 Agustus 1 Bekasi	IPA	Senior High Scool

Appendix 2 – Supporting Data from Social Media

The following is supporting data in the form of testimonials or reviews of users of electronic auction platforms related to problems studied through social media.

The picture above is a review of a user of one of the online auction platforms who feels that there is manipulation during the auction process, where the user previously won the auction asset but then the auction winner changed in the last second.



The image above is a review of another user who experienced a similar case to the previous user in the image above.

Appendix 3 - Developer Guide

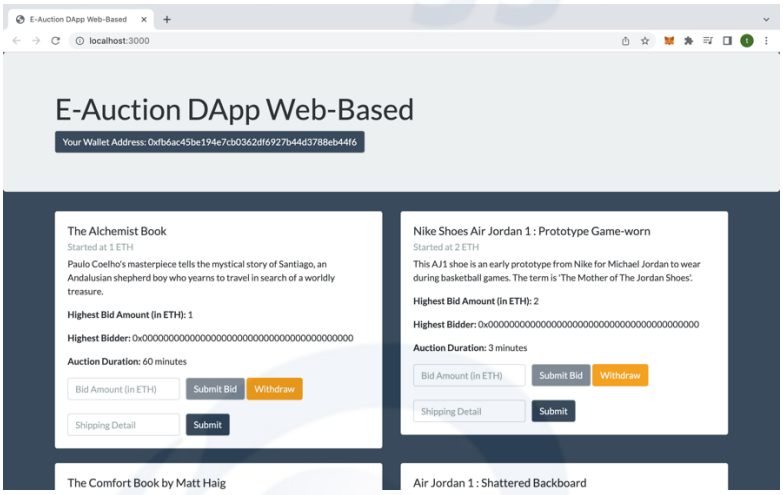
How to running an E-Auction application Using a Web-Based DApp project:

1. Open the ganache workspace app on the desktop
2. Run the "truffle migrate" command on the terminal to compile and migrate the contract
3. Run the "npm run dev" command on the terminal to run DApp
4. DApp can be run in web browsers on http://localhost:3000/
5. Import the private key on the account address in ganache to metamask E-wallet

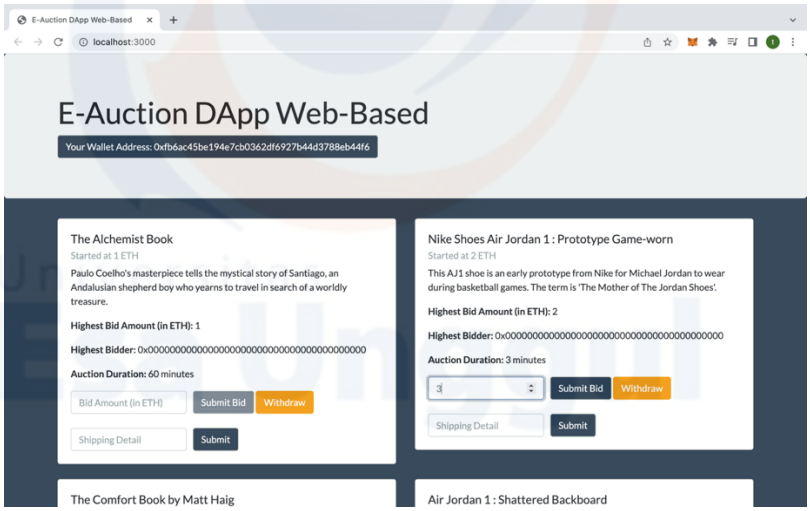
Appendix 4 – DApp User Guide

User Guide for using this E-Auction Decentralized Application:

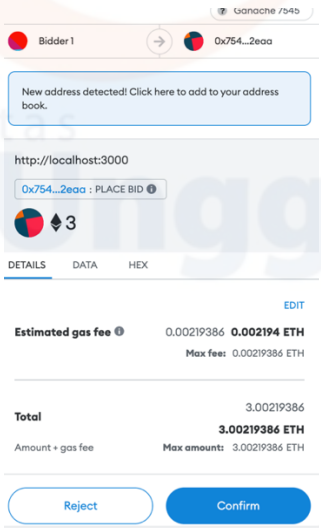
1. Connect a metamask wallet address with DApp

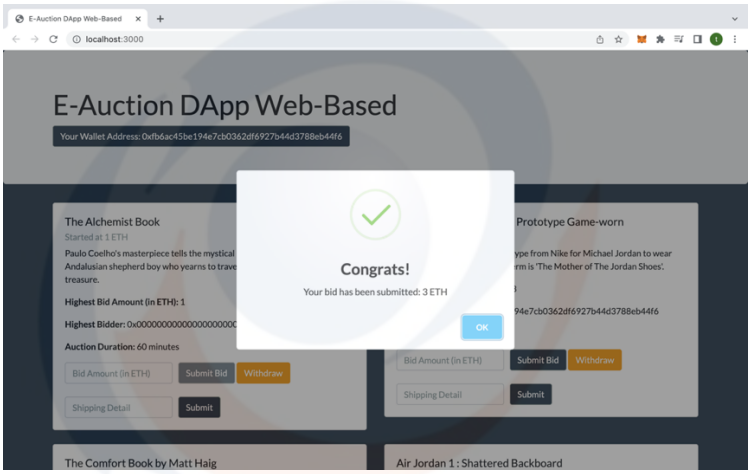


2. Enter the bid amount above the current highest bid price

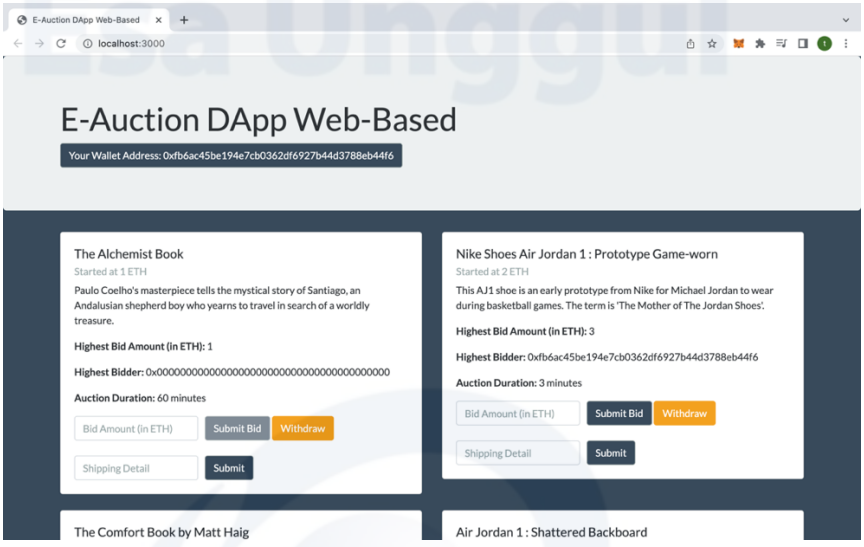


3. Confirm the transactions on metamask e-wallet, the bid amount that has been previously entered is used as the amount of funds that must be deposited by the bidder



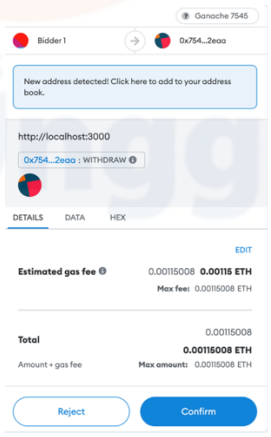


4. Highest bidder and highest bid will be updated automatically

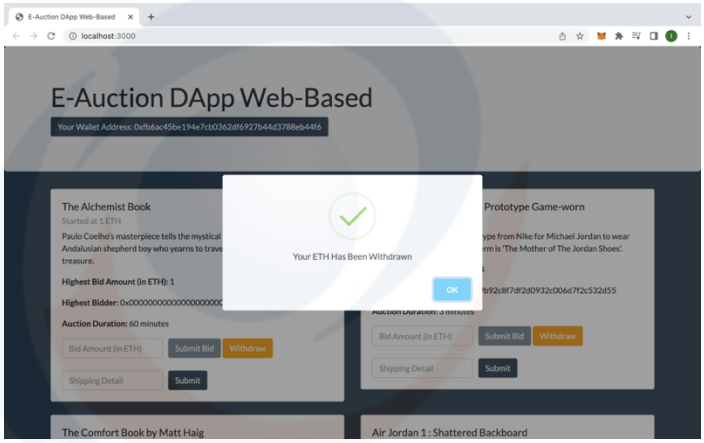


5. The losing bidder can withdraw the deposit funds

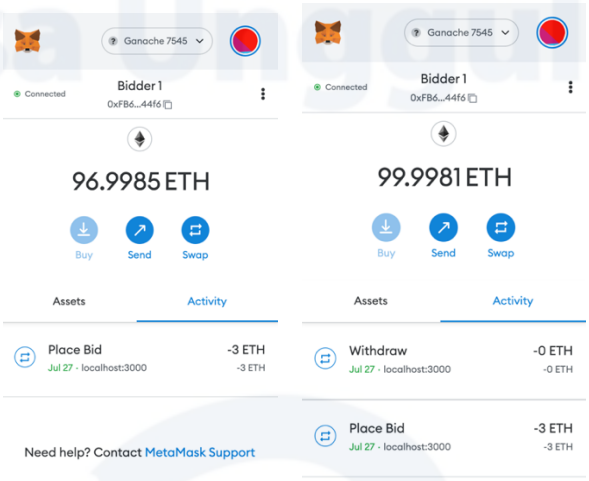
- Confirm withdrawal transaction



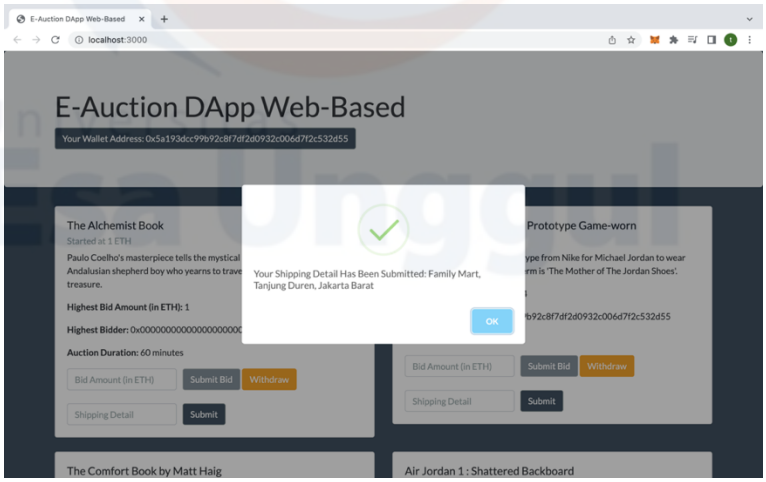
- Notification of successful withdrawal



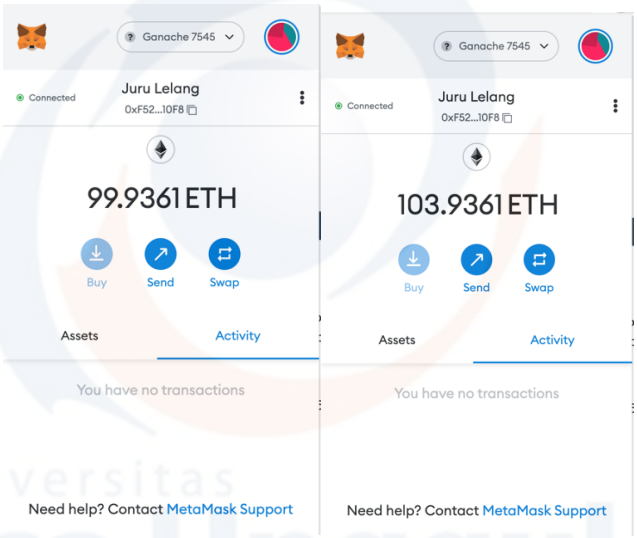
- Update balance after withdrawal transaction of deposit funds



6. The auction winner can submit shipping details as confirmation of asset delivery and release of funds to the auctioneer



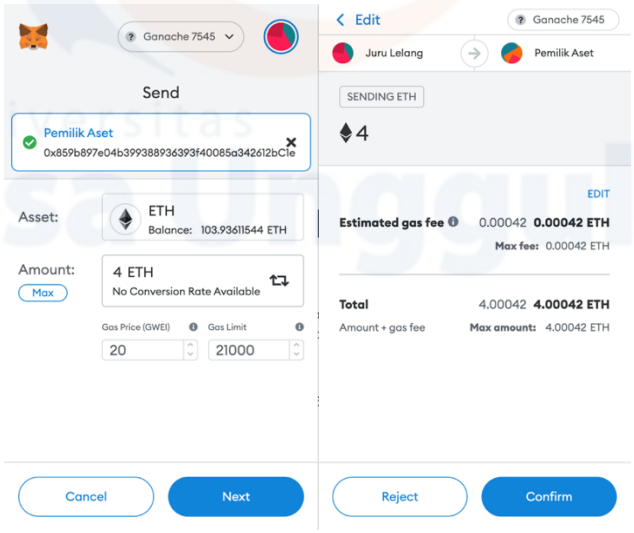
7. The auctioneer's account address automatically receive fund according to the highest bid of the asset



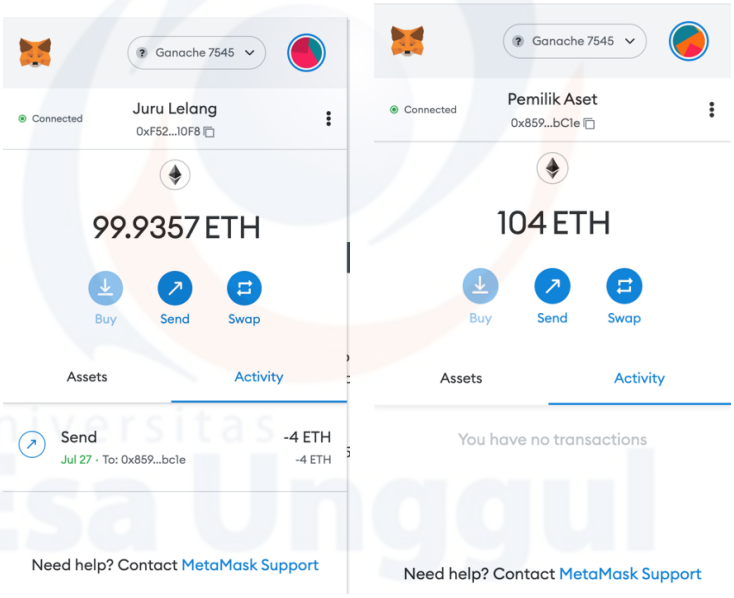
It is stated on the wallet that the balance on the auctioneer's account address is automatically increased by 4 ether in accordance with the highest bid entered by bidder 2 as the winner of the auction in the test in CHAPTER 4.

8. The auctioneer forwards the winning funds of the auction to the asset owner

- Confirmation of transactions on metamask



- Updates the balances of auctioneers and asset owners



Appendix 5 – Source Code (Auction.sol)

```
// SPDX-License-Identifier
pragma solidity ^0.5.0;

contract Auction {

    //Item Array
    struct Item{
        string item_name; // name
        string item_desc; // description
        uint base_price; // start price
        uint auction_price;  // current price of item
        uint auctionDuration;
        string shippingDetail;
    }


    // global variabel to count the auction
    uint constant itemCount = 4;
    uint[itemCount] public arrayForItems;
    uint public itemId = 0;

    // mapping pending return so the bidder can withdraw their eth
    mapping(address => mapping(uint=>uint)) private pendingReturns;

    // auction item mapping
    mapping(uint => Item) public items;

    // highest bidder mapping
    mapping(uint => address) public highestBidders;



    // to trigger bid function everytime user place a bid
    event BidEvent(uint _itemId, uint indexed _bidAmt);


    // (price convert from wei to ether) 1 ether = 1.000.000.000.000.000.000 wei
    constructor() public {
        addItem("The Alchemist Book", "Paulo Coelho's masterpiece tells the mystical story of
Santiago, an Andalusian shepherd boy who yearns to travel in search of a worldly treasure.",
10000000000000000, 10000000000000000,60, "");
        addItem("Nike Shoes Air Jordan 1 : Prototype Game-worn", "This AJ1 shoe is an early
prototype from Nike for Michael Jordan to wear during basketball games. The term is 'The Mother of
The Jordan Shoes'.", 200000000000000000,  200000000000000000,3, "");
        addItem("The Comfort Book by Matt Haig", "The new uplifting book from Matt Haig, the New
York Times bestselling author of The Midnight Library, for anyone in search of hope, looking for a
path to a more meaningful life, or in need of a little encouragement.", 300000000000000000,
300000000000000000,5, "");
        addItem("Air Jordan 1 : Shattered Backboard", "shoes worn by Michael Jordan in a friendly
match hosted by Nike. These shoes were used in a moment that was both shocking and amazing when MJ
did a slam dunk hard enough to break the backboard ring. These shoes become memories for
unforgettable moments for every basketball fan who witnessed at the time.", 500000000000000000,
500000000000000000, 10, "");
    }

    //startPrice = current auctionPrice
    // (itemCount starts at 0)
    function addItem (string memory _name, string memory _desc, uint _baseValue,  uint
_auctionPrice, uint _auctionDuration, string memory _shippingDetail) private {
        items[itemId] = Item(_name, _desc, _baseValue, _auctionPrice, (block.timestamp +
(_auctionDuration*60)), _shippingDetail);
        highestBidders[itemId] = address(0);
        itemId ++;
    }



    // Function to get item count
    function getItemCount () public pure returns (uint) {
        return itemCount;
    }


    // Function to get the name
```

```
    function getItemName (uint _itemId) public view returns (string memory) {
        require(_itemId >= 0 && _itemId < itemCount, "Item does not exist"); // the item id must be
greater than 0 but less or equal to the total count

        return items[_itemId].item_name;
    }

        // Function to get auction duration
    function getAuctionDuration (uint _itemId) public view returns (uint) {
        require(_itemId >= 0 && _itemId < itemCount, "Item does not exist"); // the item id must be
greater than 0 but less or equal to the total count

        return items[_itemId].auctionDuration;
    }



    // Function to get the highest current bid
    function getItemPrice (uint _itemId) public view returns (uint) {
        require(_itemId >= 0 && _itemId < itemCount, "Item does not exist"); // the item id must be
greater than 0 but less or equal to the total count

        return items[_itemId].auction_price;
    }



    // Function to get percent increase in value over original listing price
    function getPercentIncrease (uint _itemId) public view returns (uint) {
        uint auctionPrice = items[_itemId].auction_price;
        uint basePrice = items[_itemId].base_price;
        uint percentIncrease = (auctionPrice - basePrice)*100/basePrice;

        return percentIncrease;
    }


    // Function to get numerical information for all items in the auction as an array
    function getArrayOfNumericalInformation (uint num) public view returns (uint[itemCount] memory)
{
        uint[itemCount] memory arrayOfNumbers;

        for (uint i=0;i < itemCount; i++) {
            if (num == 1) {
                arrayOfNumbers[i] = this.getItemPrice(i);
            } else if (num == 2) {
                arrayOfNumbers[i] = this.getPercentIncrease(i);
            }
        }

        return arrayOfNumbers;
    }



    // Function to get array of prices of all items in auction as an array
    function getArrayOfPrices () public view returns (uint[itemCount] memory) {
        return this.getArrayOfNumericalInformation(1);
    }



    // Function to get array of increase in percentages of all items in auction as an array
    function getArrayOfIncreases () public view returns (uint[itemCount] memory) {
        return this.getArrayOfNumericalInformation(2);
    }



    // Function to get array of increments of all items in auction as an array
    function getArrayOfIncrements () public view returns (uint[itemCount] memory) {
        return this.getArrayOfNumericalInformation(3);
    }

    // function to get pending returns
    function getPendingReturns(address _bidder,uint _itemId) external view returns(uint){
        return pendingReturns[_bidder][_itemId];
    }

    //function get highest bidder (not in array)
    function getHighestBidder(uint _itemId) public view returns(address) {
      return highestBidders[_itemId];
```

```
    }

    // Function to get the array of highest bidders
    function getHighestBidders () public view returns (address[itemCount] memory) {
        address[itemCount] memory arrayOfBidders;

        for (uint i=0;i < itemCount; i++) {
            arrayOfBidders[i] = highestBidders[i];
        }

        return arrayOfBidders;

    }


    // Function to place a bid
    function placeBid (uint _itemId) public payable returns (uint) {

        uint _bidAmt = msg.value;
        // Requirements
        require(_itemId >= 0 && _itemId < itemCount, "Bidding on an invalid item");

        require(block.timestamp < getAuctionDuration(_itemId), "the auction had ended");

        require(check_bid (_itemId, _bidAmt),"Bid is lower or equal to the highest bid value");

        require(check_highest_bidder(_itemId, msg.sender), "Person bidding is the highest bidder");

        items[_itemId].auction_price = _bidAmt;
        highestBidders[_itemId] = msg.sender;

         //funds mapping so the user can get his money back when he lost
        if (_bidAmt != 0){
           pendingReturns[msg.sender][_itemId] += _bidAmt;
        }

        emit BidEvent(_itemId, _bidAmt);

        return _itemId; // return the item back

    }



    // Function to check if the bid is greater than highest bid
    function check_bid (uint _itemId, uint _bidAmt) public view returns (bool) {
        if (_bidAmt > items[_itemId].auction_price) return true;
        else return false;
    }



    // Function to check if person bidding is the highest bidder
    function check_highest_bidder (uint _itemId, address person_wallet) public view returns (bool)
{
        if (person_wallet == highestBidders[_itemId]) {
            return false;
        } else {
            return true;
        }
    }

    // function to withdraw eth
    function withdraw(uint _itemId) public payable returns(bool){
        uint amount = pendingReturns[msg.sender][_itemId];
        address winner = highestBidders[_itemId];


        if(msg.sender == winner){
            pendingReturns[msg.sender][_itemId] = 0;
            revert("you are the winner of this auction");
        }

        // make the pending returns to 0 everytime user click withdraw (avoid DAO)
        if(amount > 0){
            pendingReturns[msg.sender][_itemId] = 0;

            //if fail to sending money back
            if(!(msg.sender).send(amount)){
```

```
                pendingReturns[msg.sender][_itemId] = amount; //the money will back to the amount
container
                return false;
            }
        }
        return true;
    }

     // stored shipping detail
    function submitShippingDetail(uint _itemId, string memory _shippingDetail) public returns(bool)
{

        uint winnerAmount = items[_itemId].auction_price;

        // init address juru lelang
        address payable auctioneerAddress = 0xF528bb55c1c0BE10c23D5b311F6cd652116B10F8;


        require(block.timestamp >= getAuctionDuration(_itemId), "the auction not ended yet");

        if (msg.sender == getHighestBidder(_itemId)){
              // shippingDetail = _shippingDetail;
            items[_itemId].shippingDetail = _shippingDetail;

            if(winnerAmount > 0){
                items[_itemId].auction_price = 0;

                //if fail to sending money back
                if(!(auctioneerAddress).send(winnerAmount)){
                    items[_itemId].auction_price = winnerAmount; //the money will back to the
amount container
                }
                return false;

            }
            return true;
        }
        else{
            revert("you are not the winner of this auction");
        }

    }

    // return value of shipping detail
    function getShippingDetail(uint _itemId) public view returns (string memory){

        return items[_itemId].shippingDetail;
    }


}
```

Appendix 6 – Source Code (app.js)

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',

  init: async function () {
    // Load auction.
    $.getJSON('../aucD.json', function (data) {
      var auctionRow = $('#auctionRow');
      var auctionTemplate = $('#auctionTemplate');

      for (i = 0; i < data.length; i++) {
        // Adding data from json
        auctionTemplate.find('.panel-title').text(`Auction ${i + 1}`);
        auctionTemplate.find('.auction-name').text(data[i].name);
        auctionTemplate.find('.auction-description').text(data[i].description);
        auctionTemplate
          .find('.base-price')
          .text(`Started at ${data[i].original_price} ETH`); // base price

        auctionTemplate
          .find('.auction-duration')
          .text(`${data[i].auctionDuration}`);

        // Creating identifier attributes for HTML elements
        auctionTemplate.find('.shipping-detail').attr('data-id', data[i].id);
        auctionTemplate.find('.highest-bid').attr('data-id', data[i].id);

        // adding attribute to associate itemids to submit buttons
        auctionTemplate.find('.btn-submit').attr('data-id', data[i].id);
        auctionTemplate.find('.btn-withdraw').attr('data-id', data[i].id);
        auctionTemplate
          .find('.btn-shipping-detail')
          .attr('data-id', data[i].id);

        //form field
        auctionTemplate
          .find('.input-amount')
          .attr('id', `input-amt-${data[i].id}`);
        auctionTemplate
          .find('.input-shipping-detail')
          .attr('id', `input-shp-${data[i].id}`);

        auctionRow.append(auctionTemplate.html());
      }
    });

    return await App.initWeb3();
  },

  // Function to intialize web3
  initWeb3: async function () {
    if (window.ethereum) {
      App.web3Provider = window.ethereum;
      try {
        window.ethereum.autoRefreshOnNetworkChange = false;
        // Request account access
        await window.ethereum.request({ method: 'eth_requestAccounts' });
      } catch (error) {
        // User denied account access...
        console.error('User denied account access');
      }
    }
    // Legacy dapp browsers...
    else if (window.web3) {
      App.web3Provider = window.web3.currentProvider;
    }
    // If no injected web3 instance is detected, fall back to Ganache
    else {
      App.web3Provider = new Web3.providers.HttpProvider(
        'http://localhost:7545'
      );
    }
    web3 = new Web3(App.web3Provider);
```

```
    return App.initContract();
  },

  // Function to initialize contract
  initContract: function () {
    $.getJSON('Auction.json', function (data) {
      // Get the necessary contract artifact file and instantiate it with truffle-contract
      var AuctionArtifact = data;
      App.contracts.Auction = TruffleContract(AuctionArtifact);

      // Set the provider for the contract
      App.contracts.Auction.setProvider(App.web3Provider);

      // Set up the accounts
      web3.eth.getCoinbase(function (err, account) {
        if (err === null) {
          App.account = account;
          $('#account').text(account);
        }
      });

      return App.updateAuctionPrices();
    });

    return App.bindEvents();
  },

  bindEvents: function () {
    $(document).on('click', '.btn-submit', App.handleBid);
    $(document).on('click', '.btn-withdraw', App.handleWithdraw);
    $(document).on('click', '.btn-shipping-detail', App.handleShippingDetail);
  },

  //boolean changes in bid button
  handleInputChanges: function (id, bidAmount2) {
    var account = App.account;
    var aucId = id.split('-')[2];

    console.log(id);
    var highestBidder = $(document).find('.highest-bidder').eq(aucId).text();

    var highestBid2 = Number(
      $(document).find(`.highest-bid[data-id=${aucId}]`).text()
    );

    var highestBid = web3.fromWei(highestBid2, 'ether');
    var bidAmount = web3.fromWei(bidAmount2, 'ether');

    console.log('hb' + highestBid);
    console.log('ba' + bidAmount);

    if (account !== highestBidder) {
      if (bidAmount > highestBid) {
        $(document)
          .find(`.btn-submit[data-id=${aucId}]`)
          .prop('disabled', false);
      } else {
        $(document)
          .find(`.btn-submit[data-id=${aucId}]`)
          .prop('disabled', true);
      }
    } else {
      $(document).find(`.btn-submit[data-id=${aucId}]`).prop('disabled', true);
    }
  },

  updateAuctionIncreases: function () {
    var auctionInstance;

    App.contracts.Auction.deployed()
      .then(function (instance) {
        auctionInstance = instance;

        return auctionInstance.getArrayOfIncreases.call();
      })
      .then(function (increases) {
        for (j = 0; j < increases.length; j++) {
          $(document).find('.incr-in-value').eq(j).text(`${increases[j]}%`);
        }
```

```
    })
    .catch(function (err) {
      console.log(err.message);
    });
},

updateAuctionPrices: function () {
  var auctionInstance;

  App.contracts.Auction.deployed()
    .then(function (instance) {
      auctionInstance = instance;

      return auctionInstance.getArrayOfPrices.call();
    })
    .then(function (result) {
      for (j = 0; j < result.length; j++) {
        $(document)
          .find('.highest-bid')
          .eq(j)
          .text(`${web3.fromWei(result[j], 'ether')}`);
        // web3.fromWei(result[j], 'ether');
      }
    })
    .then(function (result) {
      return App.updateAuctionIncreases();
    })
    .then(function (result) {
      return App.updateHighestBidders();
    })
    .catch(function (err) {
      console.log(err.message);
    });
},

updateHighestBidders: function () {
  var auctionInstance;

  App.contracts.Auction.deployed()
    .then(function (instance) {
      auctionInstance = instance;

      return auctionInstance.getHighestBidders.call();
    })
    .then(function (bidders) {
      for (j = 0; j < bidders.length; j++) {
        $(document).find('.highest-bidder').eq(j).text(`${bidders[j]}`);
      }
    })
    .catch(function (err) {
      console.log(err.message);
    });
},

handleBid: function (event) {
  event.preventDefault();

  var aucId = parseInt($(event.target).data('id'));
  var bid_amount = parseInt($(`#input-amt-${aucId}`).val());

  var bid = web3.toWei(bid_amount, 'ether');
  console.log(bid);

  var auctionInstance;
  var account = App.account;

  App.contracts.Auction.deployed()
    .then(function (instance) {
      auctionInstance = instance;

      // Execute place bid as a transaction by sending account
      return auctionInstance.placeBid(aucId, {
        value: bid,
        from: account,
      });
    })
    .then(function (result) {
      return App.updateAuctionPrices();
    })
```

```
    .then(function () {
      $(`#input-amt-${aucId}`).val('');
      swal(
        'Congrats!',
        'Your bid has been submitted: ' + bid_amount + ' ETH',
        'success'
      );
    })
    .catch(function (err) {
      console.log(err.message);
    });
},

handleWithdraw: function (event) {
  event.preventDefault();

  var aucId = parseInt($(event.target).data('id'));
  console.log('aucId' + aucId);

  var auctionInstance;
  var account = App.account;

  var highestBidder = $(document).find('.highest-bidder').eq(aucId).text();

  if (account !== highestBidder) {
    App.contracts.Auction.deployed()
      .then(function (instance) {
        auctionInstance = instance;

        return auctionInstance.withdraw(aucId, {
          from: account,
        });
      })
      .then(function () {
        swal('', 'Your ETH Has Been Withdrawn', 'success');
      })
      .catch(function (err) {
        console.log(err.message);
      });
  } else {
    swal(' ', 'You Are The Highest Bidder', 'error');
  }
},

handleShippingDetail: function (event) {
  event.preventDefault();

  var aucId = parseInt($(event.target).data('id'));
  var shp = $(`#input-shp-${aucId}`).val();
  var highestBidder = $(document).find('.highest-bidder').eq(aucId).text();

  var auctionInstance;
  var account = App.account;

  if (account == highestBidder) {
    App.contracts.Auction.deployed()
      .then(function (instance) {
        auctionInstance = instance;

        let highestBidder = auctionInstance.getHighestBidder.call(aucId);
        console.log(highestBidder);

        let aucD = auctionInstance.getAuctionDuration.call(aucId);
        console.log(aucD);

        // Execute place bid as a transaction by sending account
        return auctionInstance.submitShippingDetail(aucId, shp, {
          from: account,
        });
      })
      .then(function () {
        swal(
          '',
          'Your Shipping Detail Has Been Submitted:  ' + shp,
          'success'
        );
        //get shipping detail data from sc
        let aucK = auctionInstance.getShippingDetail.call(aucId);
        console.log(aucK);
```

```
          console.log(shp);
          //disable button
          $(`#input-shp-${aucId}`).val('');
          $(document)
            .find(`.btn-shipping-detail[data-id=${aucId}]`)
            .prop('disable', true);

          //update shipping detail
          $(document).find('.shipping-detail').eq(aucId).text(`${shp}`);
          // web3.fromWei(result[j], 'ether');
        })
        .catch(function (err) {
          console.log(err.message);
        });
    } else {
      swal(
        'You Are Not The Winner',
        'Only Winner can submit the shipping detial information ',
        'warning'
      );

      $(`#input-shp-${aucId}`).val('');
    }
  },
};

$(function () {
  $(window).load(function () {
    App.init();
  });
});
```

Appendix 7 – Source Code (index.html)

```
<!DOCTYPE html>
<html lang ="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head content must come
*after* these tags -->
    <title>E-Auction DApp Web-Based</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/custom.css" rel="stylesheet">

    <!-- Font Awesome -->
    <script src="https://kit.fontawesome.com/16cdf6d73b.js" crossorigin="anonymous"></script>
    <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

  </head>
  <body class="b">
    <div>
      <div class="jumbotron feature ">
        <div class="container">
          <h1>E-Auction DApp Web-Based</h1>
          <button class="btn btn-dark ml-auto mr-1 account-button">Your Wallet Address: <span
id="account"></span></button>
        </div>
      </div>
      <div class="container">
        <div id="auctionRow" class="row justify-content">
          <!--Auction Cards loads here-->
        </div>
      </div>
    </div>
    <div id="auctionTemplate" style="display: none;">
      <div class="col-sm-6  mb-4">
        <div class="card">
          <div class="card-body">
            <h5 class="card-title auction-name">Panel title</h5>
            <h6 class="card-subtitle mb-2 text-muted base-price">Panel subtitle</h6>
            <p class="card-text auction-description">Some quick example text to build on the panel
title and make up the bulk of the panel's content.</p>
            <p class="card-text">
              <strong>Highest Bid Amount (in ETH): </strong><span class="highest-bid"></span></p>
              <strong>Highest Bidder: </strong><span class="highest-bidder"></span></p>
              <strong>Auction Duration: </strong><span class="auction-duration"></span> minutes</p>
              <form>
                <div class="form-row align-items-center">
                  <div class="col-auto">
                    <input type="number" class="form-control input-amount" placeholder="Bid Amount
(in ETH)" min="0.00" name="bid" oninput="App.handleInputChanges(this.id, this.value)">
                  </div>
                  <div class="col-auto">
                    <button type="button" class="btn btn-dark btn-submit" disabled=true >Submit
Bid</button>
                    <button class="btn btn-warning btn-withdraw" disable=true >Withdraw</button>
                  </div>
                </div>
              </form>
              <br>
              <form>
                <div class="form-row align-items-center">
                  <div class="col-auto">

                    <input type="text" class="form-control input-shipping-detail"
placeholder="Shipping Detail" name="shippingDetail" >
                  </div>
                  <div class="col-auto">
                    <button type="text" class="btn btn-dark btn-shipping-detail" disable=false
>Submit</button>
                  </div>
                </div>
              </form>
          </div>
        </div>
```

```
      </div>
    </div>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
    <script src="js/web3.min.js"></script>
    <script src="js/truffle-contract.js"></script>
    <script src="js/app.js"></script>
  </body>
</html>
```