CSCI 6313 – Assignment 1 Part B

Gitlab Repository at
https://git.cs.dal.ca/bhandari/assignment1

Aman Singh Bhandari
B00910008

Table of Contents

1. Overview of the Approach

Approach for this assignment involves a Dapp that interacts with smart contract through infura deployed on ropsten testnet. Figure 1 diagram shows the overview of the various components involved. [1]

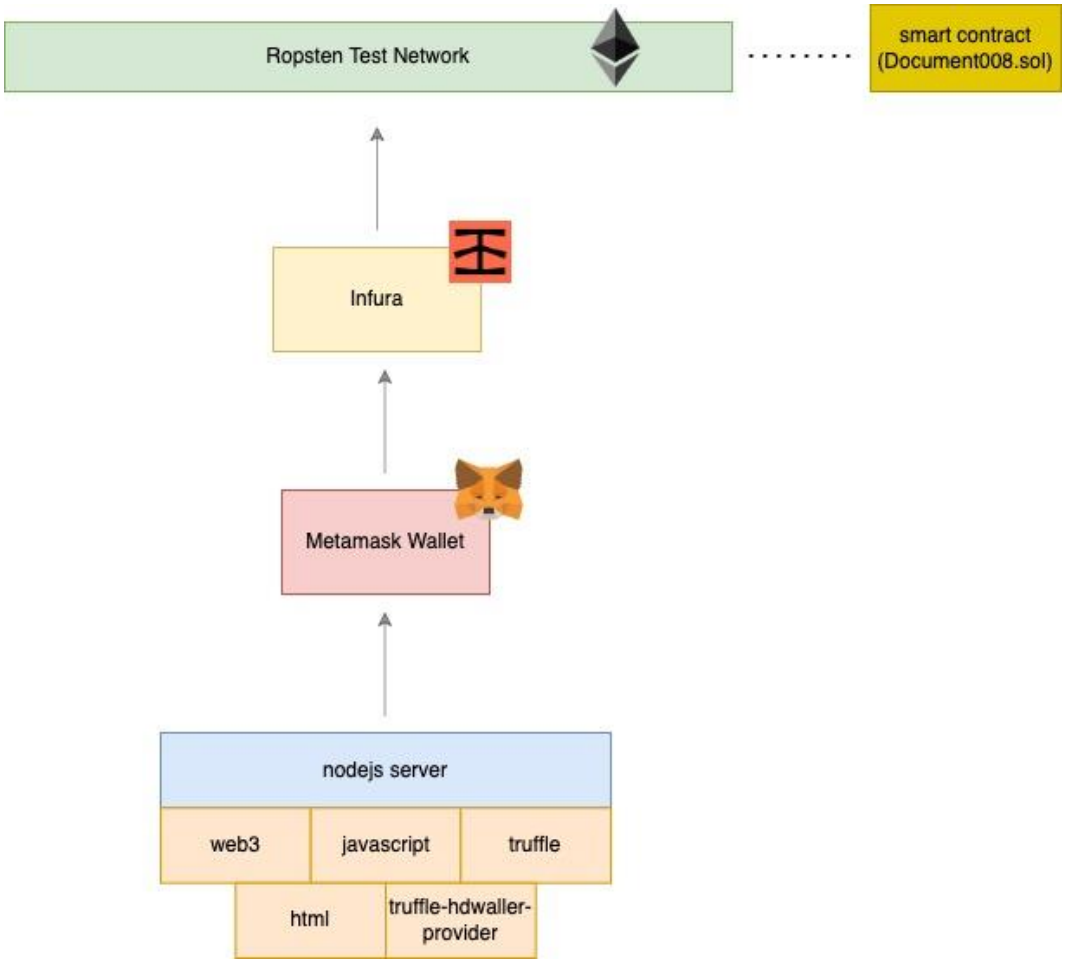

Figure 1 Various components involved in the solution

2. Algorithm of the solution

Figure 2 shows the algorithm or approach taken to act as a notary between buyer and seller to exchange valuable items.

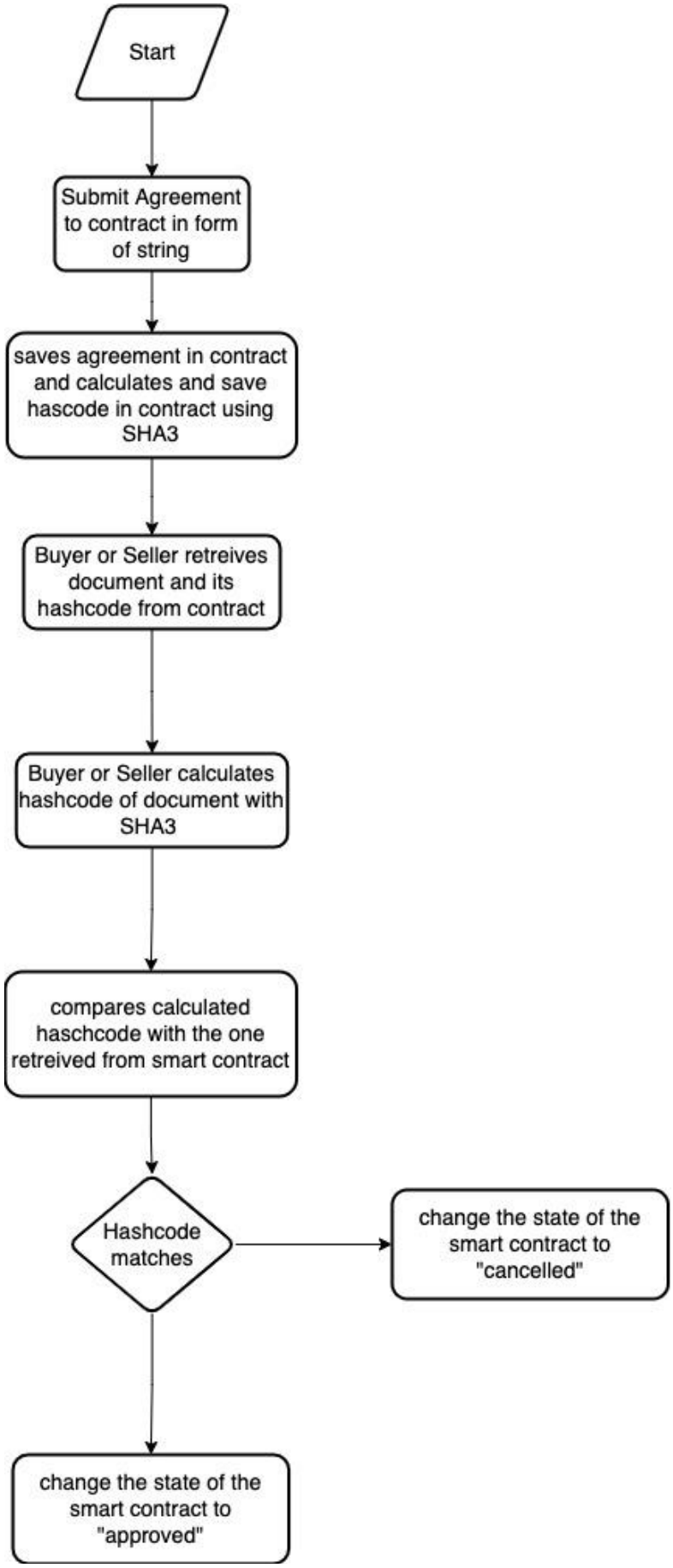

Figure 2 Algorithm of the approach taken

3. Smart contract (Document008.sol)

Figure 3 shows the smart contract that acts as a notary between buyer and seller. The contract is written in solidity. It stores the agreement document between two parties and its hashcode. Each party validates the document by verifying hash code of the document and approves/cancel the document. As you can see in the figure, It uses keccak256 in solidity to generate the hash of the document and then stores it in the contract.

Document008.sol × | index.js | truffle-config.js | Document008.json

Notary > contracts > Document008.sol

```
//SPDX-License-Identifier: Dalhousie

pragma solidity >=0.7.0 <0.9.0;

contract Document008 {
    string details = ""; //details of the agreement

    bytes32 hashCode; //hashcode of the details of the agreement

    string status = ""; //accepted or rejected

    function submitAgreement008(string memory _details) public {
        details = _details; //set the details of the agreement
        hashCode = keccak256(abi.encodePacked(_details)); //hashing using keccak256
    }

    function retrieveAgreement008()
        public
        view
        returns (
            //returns the details and its hash
            bytes32,
            string memory
        )
    {
        return (hashCode, details);
    }

    function approveAgreement008()
        public
    //changes the status of document to approved
    {
        status = "approved";
    }

    function cancelAgreement008()
        public
    //changes the status of document to cancelled
    {
        status = "cancelled";
    }
}
```

Figure 3 Smart Contract

4. Setting up Infura

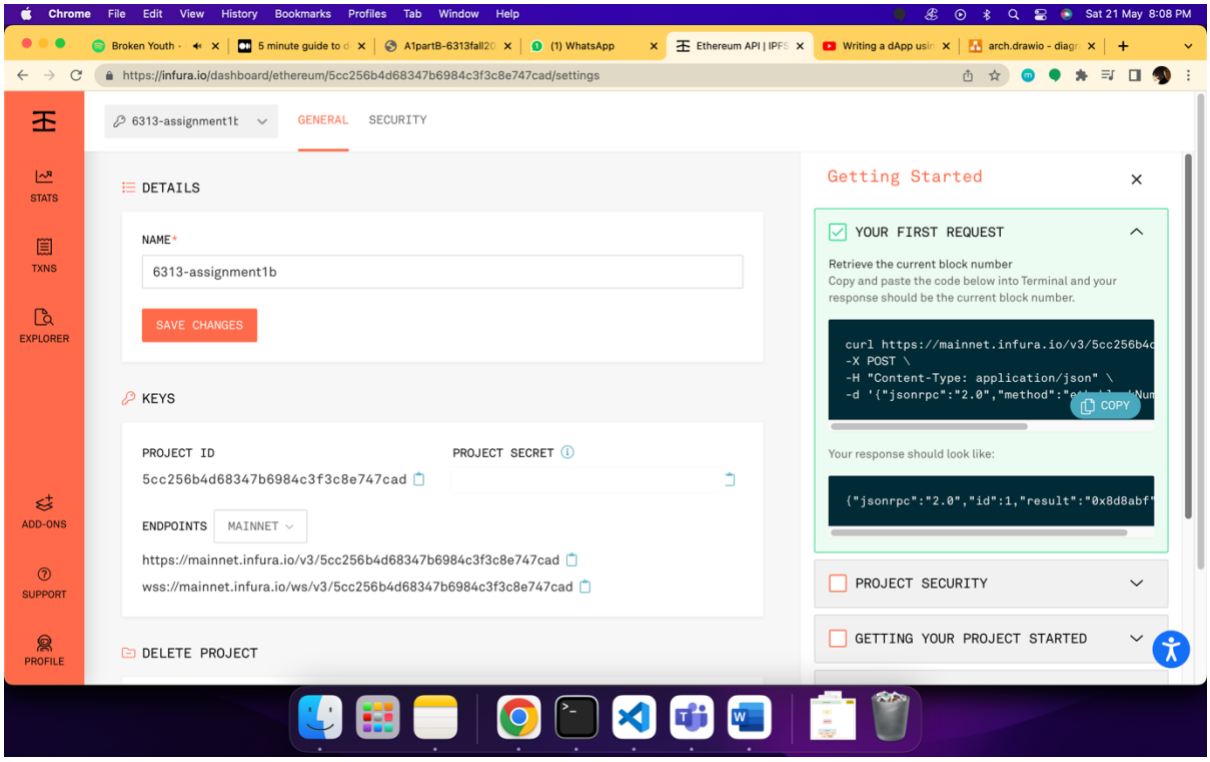

Figure 4 Infura account

5. Creating Metamask wallet (extension as well)

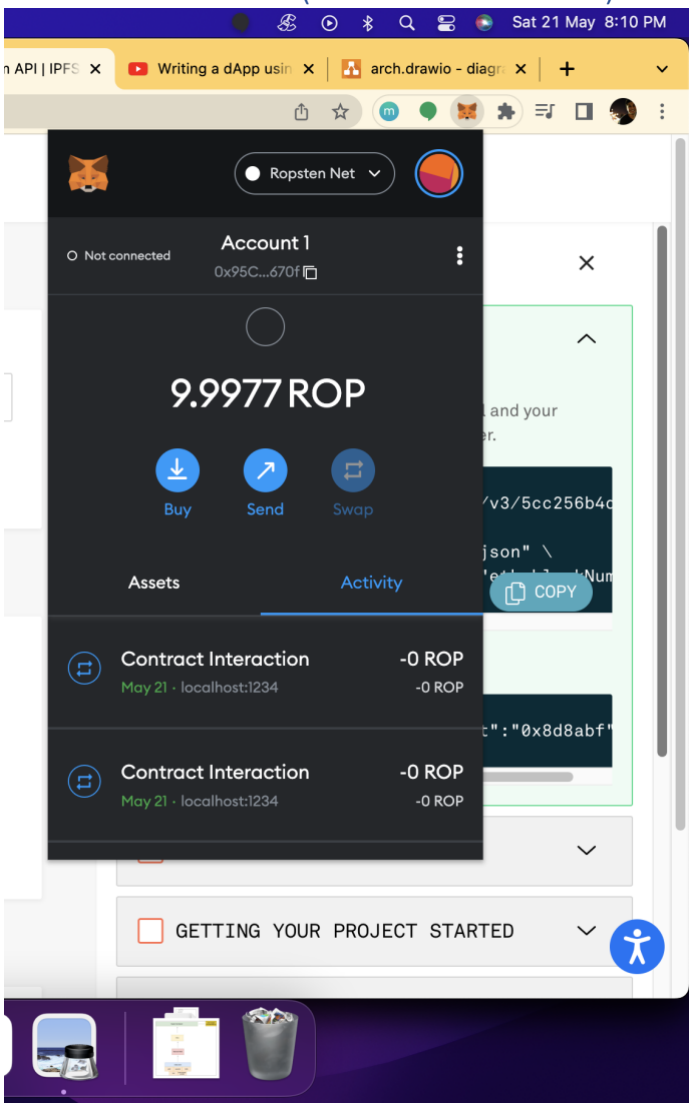

Figure 5 Metamask wallet

6. Getting test ETH for Ropsten

By providing my Ethereum address got free Ropsten testnet ETH from website https://faucet.egorfine.com/ .

7. Deploy contract to Ropsten testnet

Using command "truffle deploy --network ropsten" deployed the smart contract to testnet. Copied the address of deployed contract to be used later in javascript (web3).

8. Interacting with smart contract using web3

Figure 6 shows index.js where all the interaction with smart contract is taking place. First the account is fetched, followed by fetching contract and accessing its method.

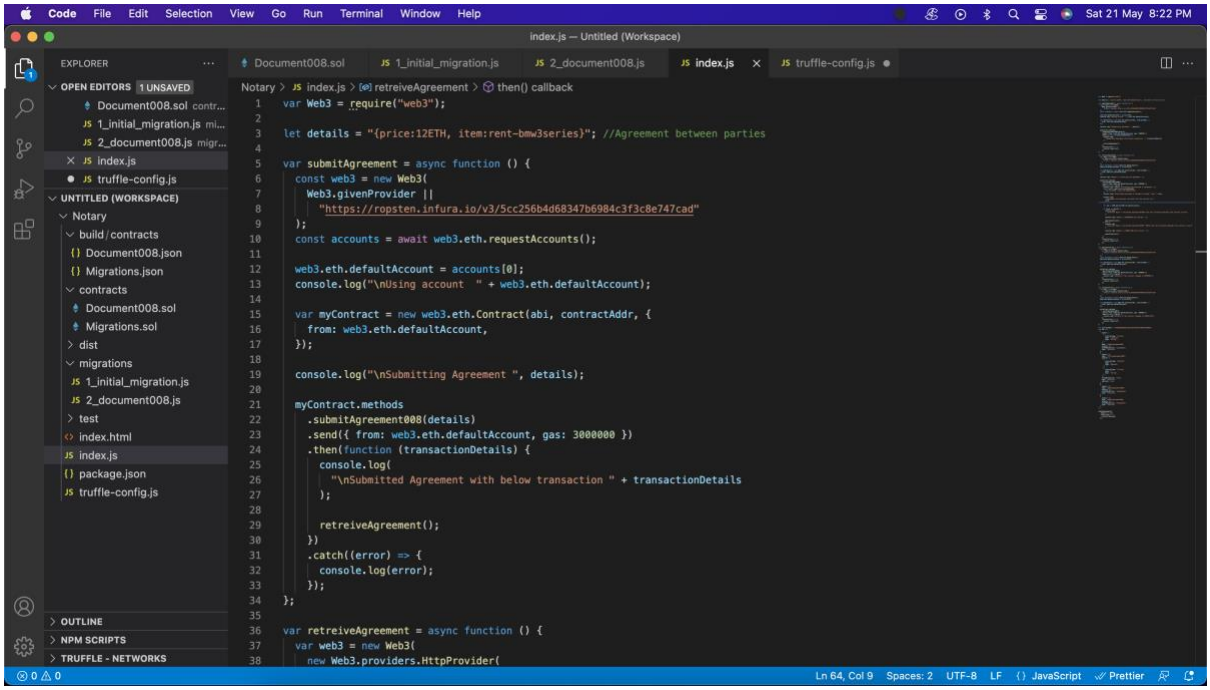

Figure 6 index.js

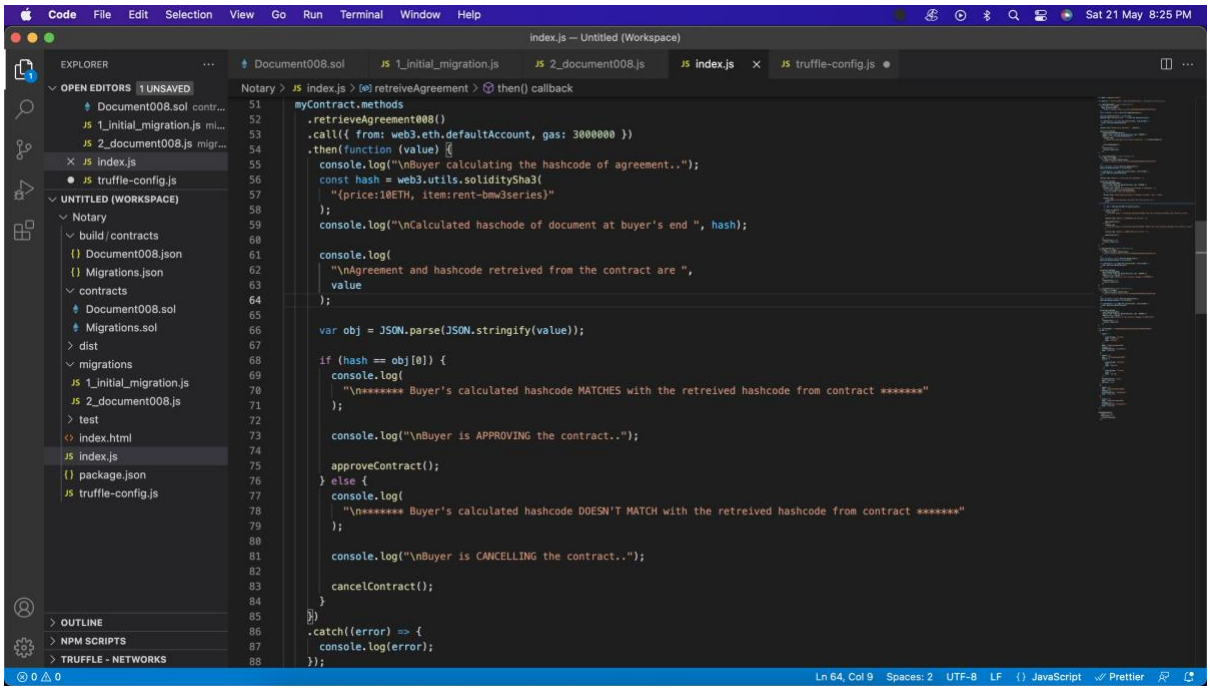

Figure 7 index.js cont..

```
var approveContract = async function () {
  var web3 = new Web3(
    new Web3.providers.HttpProvider(
      "https://ropsten.infura.io/v3/5cc256b4d68347b6984c3f3c8e747cad"
    )
  );
  const accounts = await web3.eth.getAccounts();
  web3.eth.defaultAccount = accounts[0];

  var myContract = new web3.eth.Contract(abi, contractAddr, {
    from: web3.eth.defaultAccount,
  });

  myContract.methods
    .approveAgreement008()
    .call({ from: web3.eth.defaultAccount, gas: 3000000 })
    .then(function (value) {
      console.log("\nStatus of the contract changed to APPROVED");
    })
    .catch((error) => {
      console.log(error);
    });
};

var cancelContract = async function () {
  var web3 = new Web3(
    new Web3.providers.HttpProvider(
      "https://ropsten.infura.io/v3/5cc256b4d68347b6984c3f3c8e747cad"
    )
  );
  const accounts = await web3.eth.getAccounts();
  web3.eth.defaultAccount = accounts[0];

  var myContract = new web3.eth.Contract(abi, contractAddr, {
    from: web3.eth.defaultAccount,
  });
```

Figure 8 index.js cont..

```
    .catch((error) => {
      console.log(error);
    });
};

let contractAddr = "0x0e34b8a81Bb7ce3bCCd5c84FE4cf48B47E37e6B6";
let abi = [
  {
    inputs: [
      {
        internalType: "string",
        name: "_details",
        type: "string",
      },
    ],
    name: "submitAgreement008",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function",
  },
  {
    inputs: [],
    name: "retrieveAgreement008",
    outputs: [
      {
        internalType: "bytes32",
        name: "",
        type: "bytes32",
      },
      {
        internalType: "string",
        name: "",
        type: "string",
      },
    ],
    stateMutability: "view",
    type: "function",
    constant: true,
```

Figure 9 index.js cont..

```
        name: "",
        type: "string",
      },
    ],
    stateMutability: "view",
    type: "function",
    constant: true,
  },
  {
    inputs: [],
    name: "approveAgreement008",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function",
  },
  {
    inputs: [],
    name: "cancelAgreement008",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function",
  },
];

submitAgreement()
  .then(() => {})
  .catch((msg) => {
    console.log(msg);
  });
```

Figure 10 index.js cont..

9. Output

```
http://localhost:1234
```

```
Using account  0x95CcA48c734C733b8E92fF0b8e0f9502C63d670f                                   index.js:13
Submitting Agreement  {price:10ETH, item:rent-bmw3series}                                   index.js:19
DevTools failed to load source map: Could not load content for chrome-extension://kdfieneakcjfaiglcfcgkidlkmlijjnh/content/js/bundle.tracing.min.js.map: System error: net::ERR_FILE_NOT_FOUND
DevTools failed to load source map: Could not load content for chrome-extension://kdfieneakcjfaiglcfcgkidlkmlijjnh/writer/js/angular-ui-router.js.map: System error: net::ERR_FILE_NOT_FOUND
Running Ginger Widget v2.0.189                                                              writer.vendor.js:2
Submitted Agreement with below transaction [object Object]                                  index.js:25
Buyer is retreiving the agreement..                                                         index.js:49
Buyer calculating the hashcode of agreement..                                               index.js:55
Calculated haschode of document at buyer's end  0xcd3da8368f1b64254589a23acd44e5d0c6a3f419da424b03196b6a5668a5d210   index.js:59
Agreement and hashcode retreived from the contract are  ▸Result {0: '0xcd3da8368f1b64254589a23acd44e5d0c6a3f419da424b03196b6a5668a5d210', 1: '{price:10ETH, item:rent-bmw3series}'}   index.js:61
******* Buyer's calculated hashcode MATCHES with the retreived hashcode from contract *******   index.js:69
Buyer is APPROVING the contract..                                                           index.js:73
Status of the contract changed to APPROVED                                                  index.js:188
>
```

Figure 11 Actor approved the contract

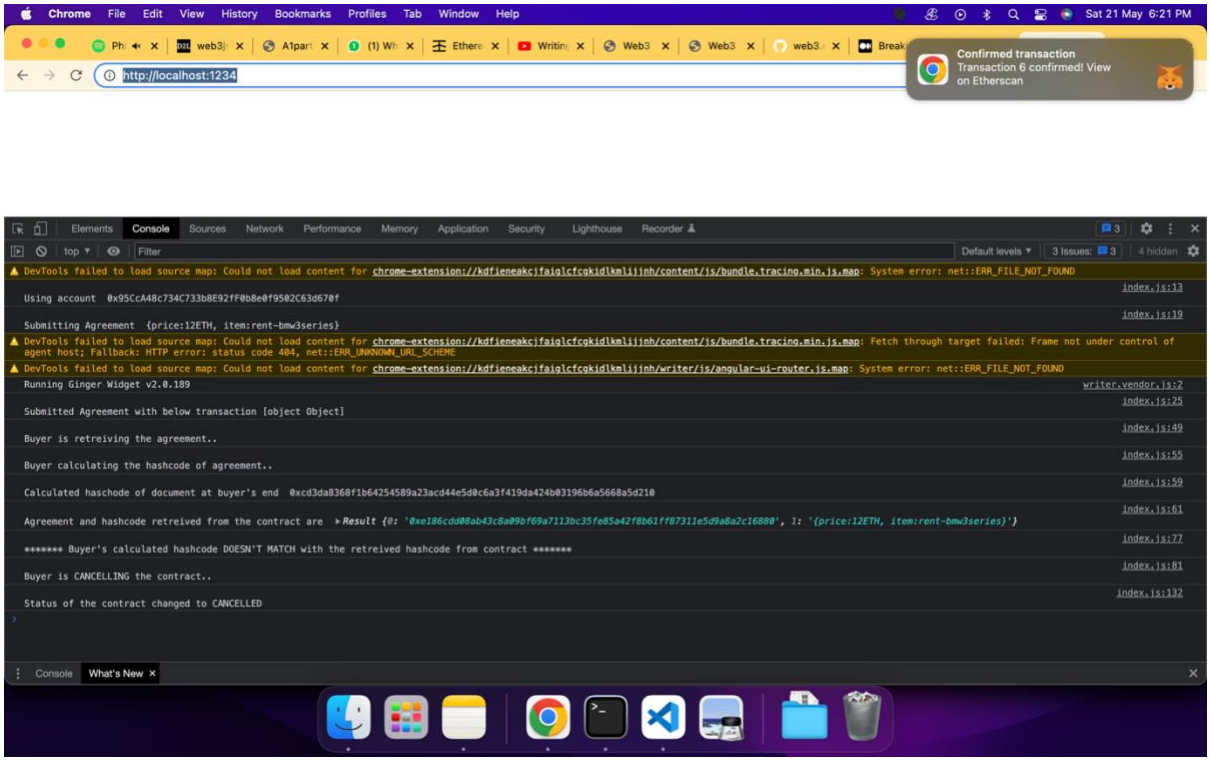

Figure 12 Actor cancelled the document

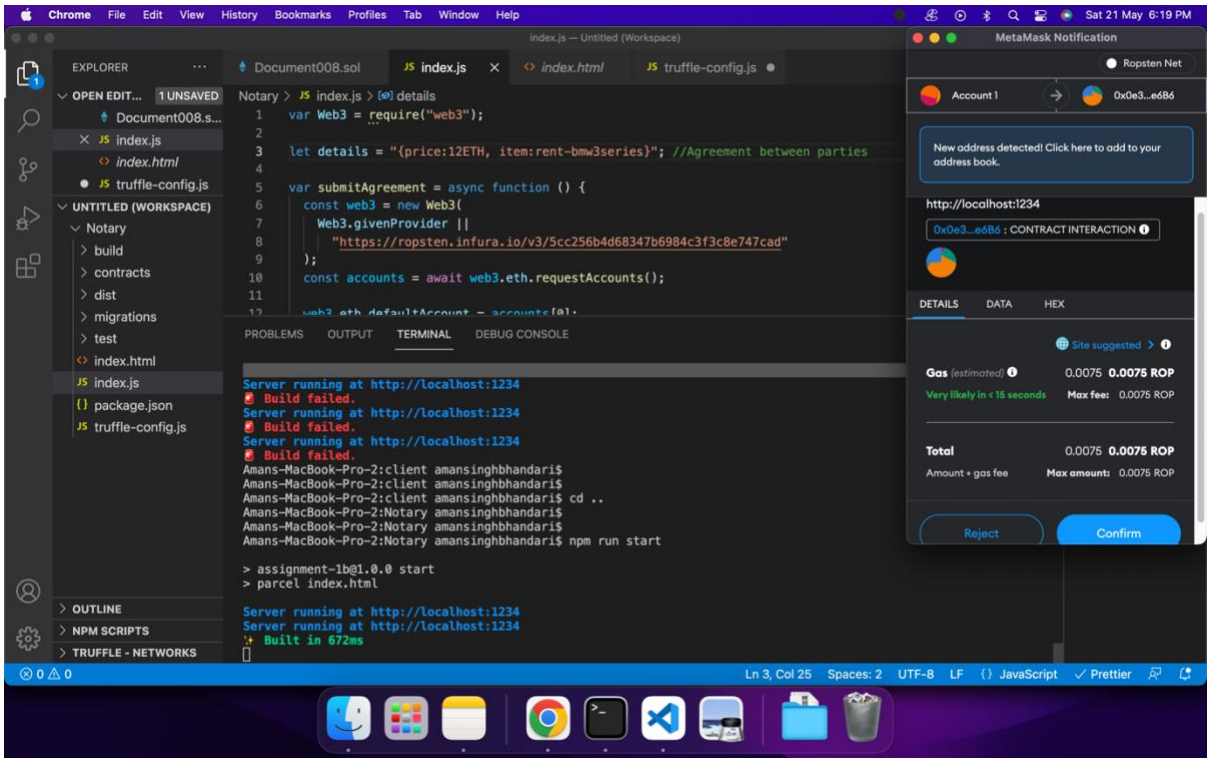

Figure 13 Metamask notification asking for permission to confirm

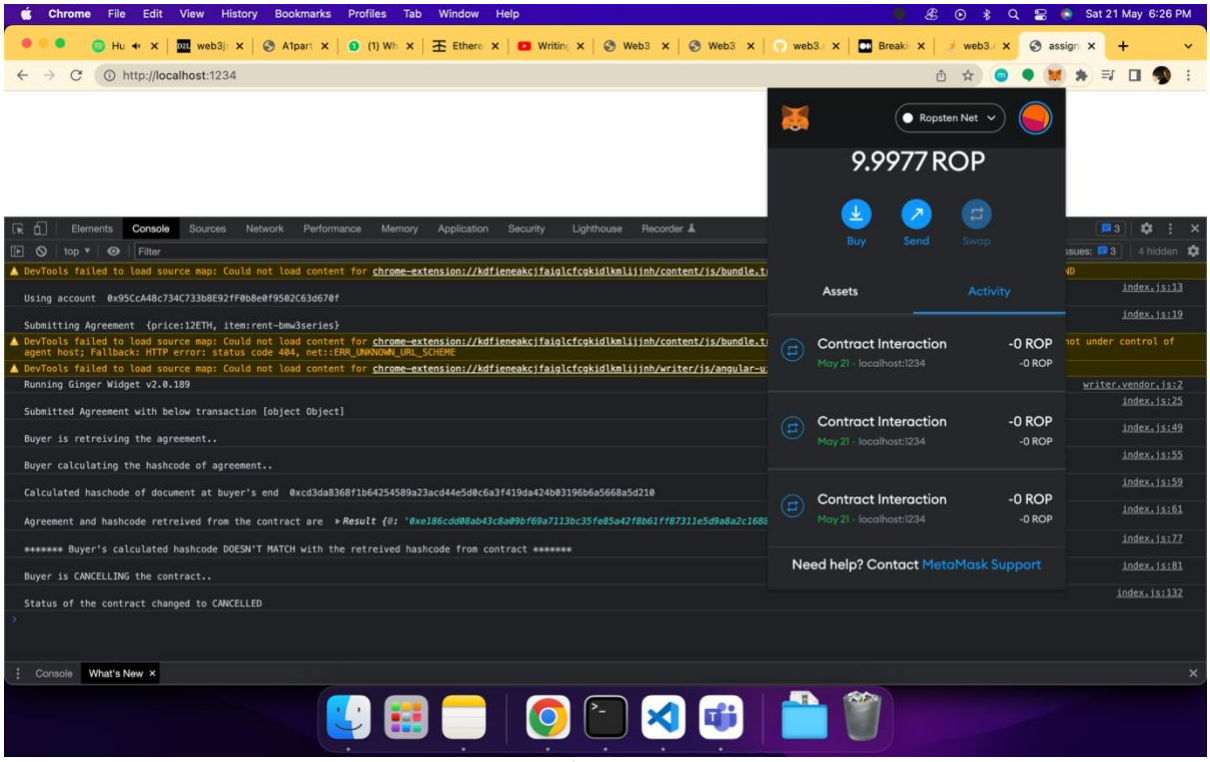

Figure 14 Contract Interaction/Transaction can be seen in MetaMask

10. References

[1] 0xSage, "5 minute guide to deploying smart contracts with Truffle and Ropsten," *Medium*, 10-May-2022. [Online]. Available: https://medium.com/coinmonks/5-minute-guide-to-deploying-smart-contracts-with-truffle-and-ropsten-b3e30d5ee1e. [Accessed: 21-May-2022].

[2] "# create a simple dapp," *MetaMask Docs*. [Online]. Available: https://docs.metamask.io/guide/create-dapp.html. [Accessed: 21-May-2022].