

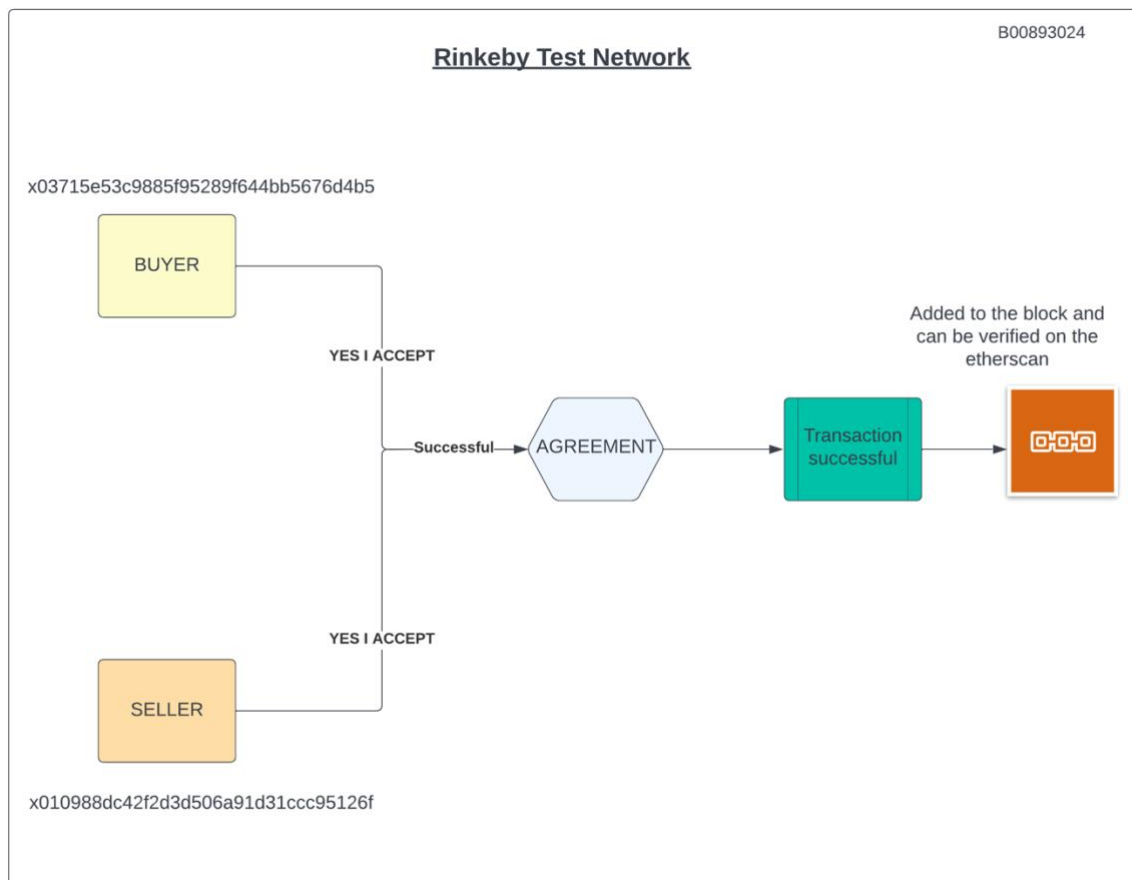
Assignment 1 - part B

Table of Contents :

1. Introduction to the Approach
2. Development Environment
3. Deploying on the Test Rinkeby Platform
4. WebUI using Parcel, Web3 and NodeJS

Introduction

The approach followed the procedural algorithm where the smart contracts act as a notary and helps in a successful transaction. Below is the flow diagram which specifies the process followed :



Environment and Assumptions

Development Environment

1. Visual Studio Code
2. Truffle CLI
3. Ganache GUI
4. MetaMask Chrome plugin
5. Rinkeby Test Ethers from TestNet
6. Etherscan
7. Infura Platform
8. Gitlab

Gitlab Link to Submission :

https://git.cs.dal.ca/ayushv/csci_6313_verma_ayush_b00893024

Screenshots :

```
> Artifacts written to /Users/ayush/Documents/University Assignments/Blockchain/Assignment_2/build/contracts
> Compiled successfully using:
  - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang
ayush@Ayush-2 Assignment_2 %
```

Figure 1 – Successful Compilation using truffle compile

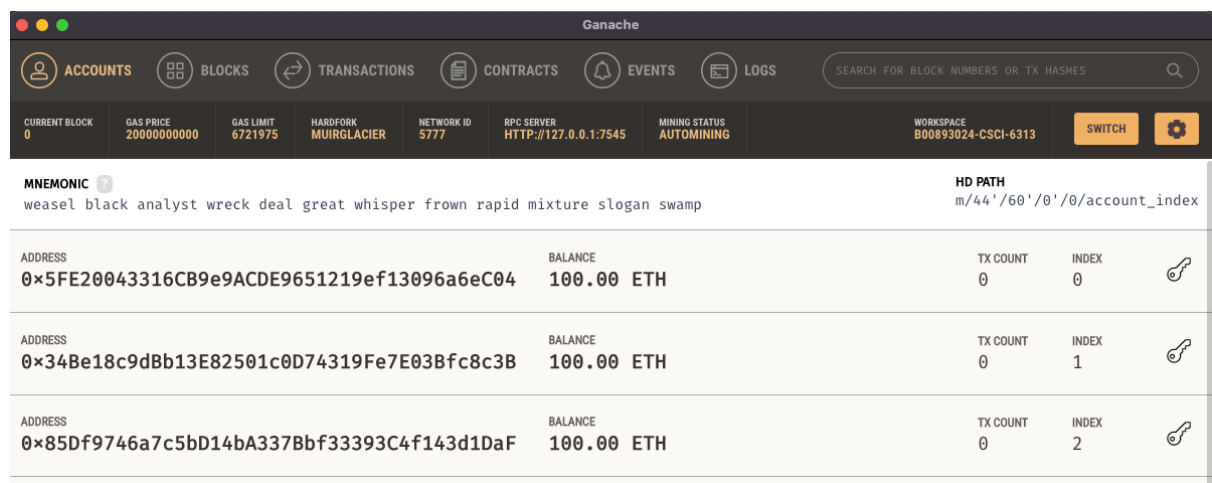


Figure 2 – Ganache GUI

```

> gas price:      3.375 gwei
> value sent:     0 ETH
> total cost:     0.00084426975 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost:     0.00084426975 ETH

2_deploy_contracts.js
=====

Deploying 'BuyerSellerNotary024'
=====
> transaction hash: 0x5f3044632fed09f744434c8dd68bc4a0e5b955030737fca4d827257b57eea96e
> Blocks: 0        Seconds: 0
> contract address: 0x0F29E2879939AF013d1BDd24414DbBcD918c0214
> block number:     3
> block timestamp:  1653161162
> account:          0x50d5b9ca4D01d2054a53776c25ebc289b786305e
> balance:          99.997642010813541673
> gas used:          428966 (0x68ba6)
> gas price:         3.178366198 gwei
> value sent:       0 ETH
> total cost:       0.001363411034491268 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost:      0.001363411034491268 ETH

Summary
=====
> Total deployments: 2
> Final cost:       0.002207680784491268 ETH

truffle(develop)>

```

Figure 3 – Local Deployment of the Smart contract-1

```

> gas price:      3.375 gwei
> value sent:     0 ETH
> total cost:     0.00084426975 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost:     0.00084426975 ETH

2_deploy_contracts.js
=====

Deploying 'BuyerSellerNotary024'
=====
> transaction hash: 0x5f3044632fed09f744434c8dd68bc4a0e5b955030737fca4d827257b57eea96e
> Blocks: 0        Seconds: 0
> contract address: 0x0F29E2879939AF013d1BDd24414DbBcD918c0214
> block number:     3
> block timestamp:  1653161162
> account:          0x50d5b9ca4D01d2054a53776c25ebc289b786305e
> balance:          99.997642010813541673
> gas used:          428966 (0x68ba6)
> gas price:         3.178366198 gwei
> value sent:       0 ETH
> total cost:       0.001363411034491268 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost:      0.001363411034491268 ETH

Summary
=====
> Total deployments: 2
> Final cost:       0.002207680784491268 ETH

truffle(develop)>

```

Figure 4 – Local Deployment of the Smart contract-2

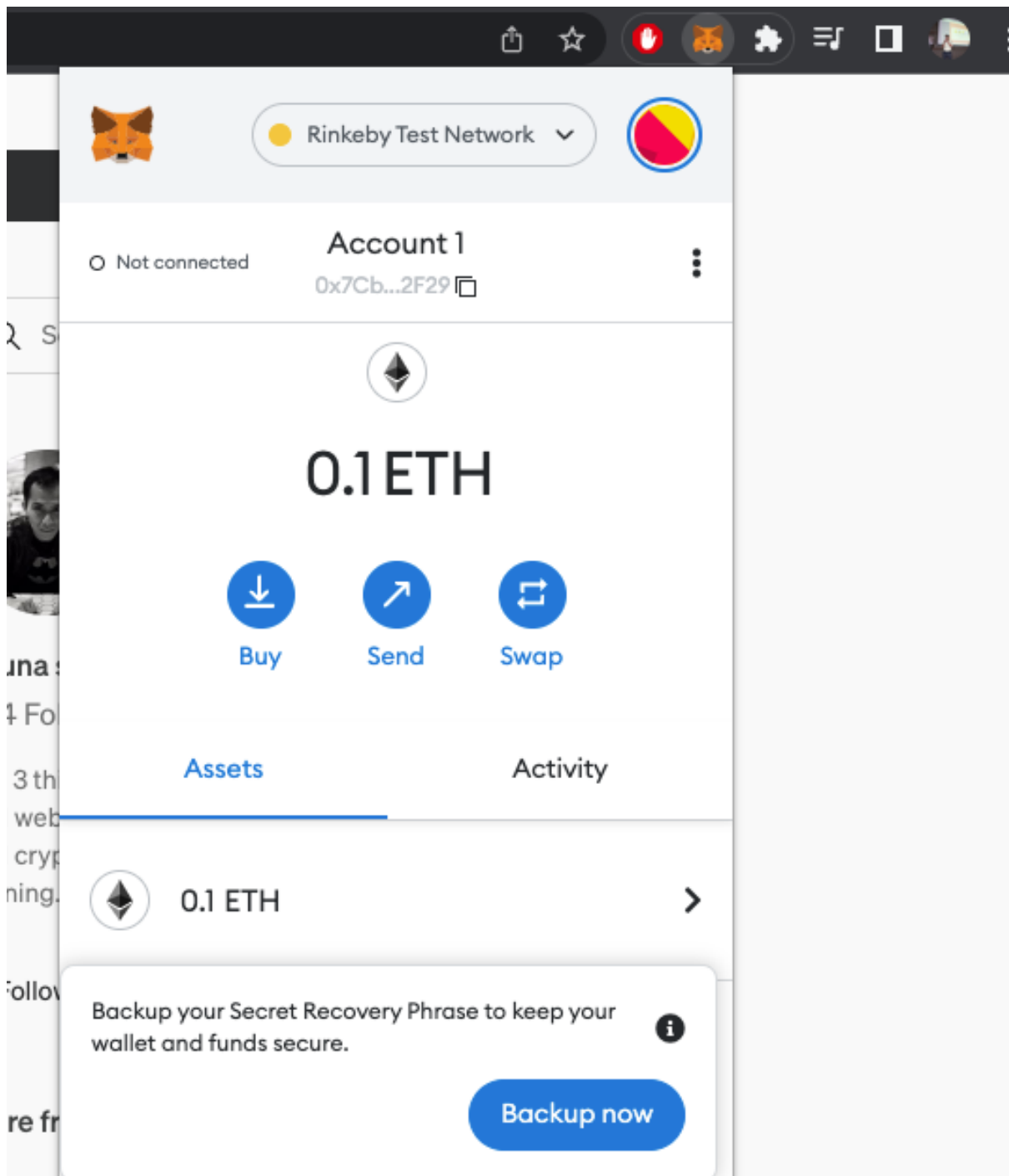


Figure 5 – Metamask wallet

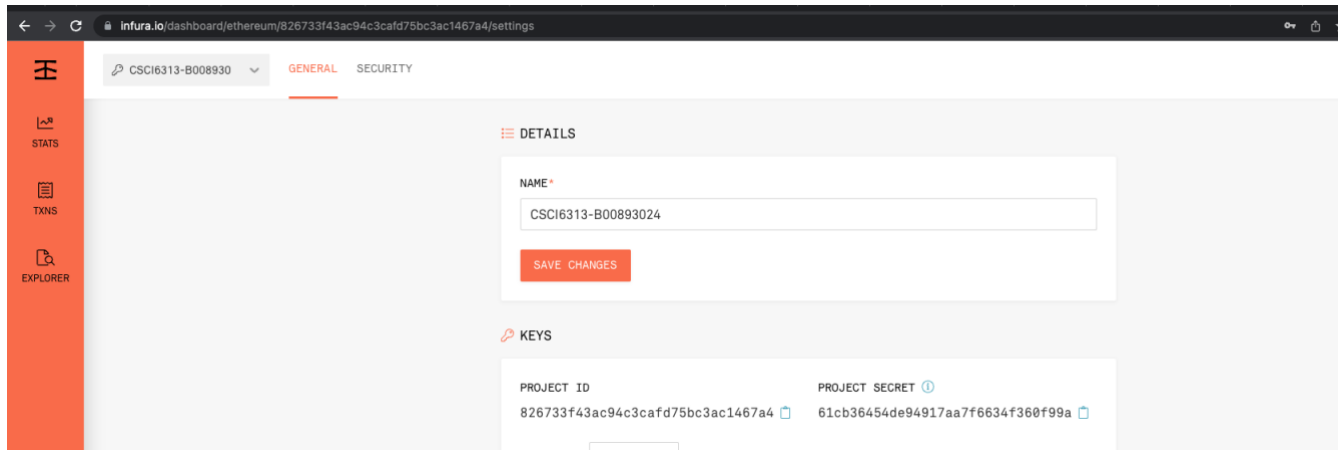


Figure 6 – Infura blockchain client workspace

```
> Compiled successfully using:
- solc: 0.8.13+commit.abaa5c0e.Emscripten.clang

Migrations dry-run (simulation)
=====
> Network name: 'rinkeby-fork'
> Network id: 4
> Block gas limit: 29999972 (0x1c9c364)

2_deploy_contracts.js
=====
Deploying 'BuyerSellerNotary024'
=====
> block number: 10716405
> block timestamp: 1653162237
> account: 0x7Cb51C6B92B7d0b5A55CbeDEcACE0f2f6B02F29
> balance: 0.098927584996568272
> gas used: 428966 (0x68ba6)
> gas price: 2.500000008 gwei
> value sent: 0 ETH
> total cost: 0.001072415003431728 ETH

> Total cost: 0.001072415003431728 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.001072415003431728 ETH
```

Figure 7 – Deploying on the Rinkeby network

```
> Final cost: 0.001072415003431728 ETH

Starting migrations...
=====
> Network name: 'rinkeby'
> Network id: 4
> Block gas limit: 29999972 (0x1c9c364)

2_deploy_contracts.js
=====
Deploying 'BuyerSellerNotary024'
=====
> transaction hash: 0x8257358907638e4bea2b53b1ae572b0bee2f6e920a59c84b4097e5f31ddf2314
> Blocks: 0 Seconds: 12
> contract address: 0xc55FCcC0cD54DFB6ae1a3766e5bC18759E8cE175
> block number: 10716410
> block timestamp: 1653162253
> account: 0x7Cb51C6B92B7d0b5A55CbeDEcACE0f2f6B02F29
> balance: 0.099528137396139306
> gas used: 428966 (0x68ba6)
> gas price: 1.100000009 gwei
> value sent: 0 ETH
> total cost: 0.000471862603860694 ETH

> Saving artifacts
=====
> Total cost: 0.000471862603860694 ETH
```

Figure 8 – Deploying on the Rinkeby network

```

=====
> Network name: 'rinkeby'
> Network id: 4
> Block gas limit: 29999972 (0x1c9c364)

2_deploy_contracts.js
=====

Deploying 'BuyerSellerNotary024'

> transaction hash: 0x8257358907638e4bea2b53b1ae572b0bee2f920a59c84b4097e5f31ddf2314
> Blocks: 0
> contract address: 0xc55fCc0cD54DFB6ae1a3766e5bC18759E8cE175
> block number: 10716410
> block timestamp: 1653162253
> account: 0x7Cb51c6892b7d0b5A55CbeDEcACE00f2f6802F29
> balance: 0.099528137396139306
> gas used: 428966 (0x68ba6)
> gas price: 1.100000009 gwei
> value sent: 0 ETH
> total cost: 0.000471862603860694 ETH

> Saving artifacts

> Total cost: 0.000471862603860694 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.000471862603860694 ETH

ayush@Ayush-2 Assignment_2 % x

```

Figure 9 – Deploying on the Rinkeby network

The screenshot shows the Etherscan interface for a transaction on the Rinkeby Testnet. The transaction is successful and has 18 block confirmations. The transaction hash is 0x8257358907638e4bea2b53b1ae572b0bee2f920a59c84b4097e5f31ddf2314. The transaction was created 4 minutes ago on May 21, 2022, at 07:44:13 PM UTC. The transaction value is 0 Ether (\$0.00). The transaction fee is 0.000471862603860694 Ether (\$0.00). The gas price is 0.00000001100000009 Ether (1.100000009 Gwei). The transaction created a contract at address 0xc55fCc0cD54DFB6ae1a3766e5bC18759E8cE175.

Field	Value
Transaction Hash	0x8257358907638e4bea2b53b1ae572b0bee2f920a59c84b4097e5f31ddf2314
Status	Success
Block	10716410 (18 Block Confirmations)
Timestamp	4 mins ago (May-21-2022 07:44:13 PM UTC)
From	0x7cb51c6892b7d0b5a55cbedecace00f2f6802f29
To	[Contract 0xc55fCc0cD54DFB6ae1a3766e5bC18759E8cE175 Created]
Value	0 Ether (\$0.00)
Transaction Fee	0.000471862603860694 Ether (\$0.00)
Gas Price	0.00000001100000009 Ether (1.100000009 Gwei)

Figure 10 – Etherscan transaction

Transaction Details < >

Overview State

[This is a Rinkeby Testnet transaction only]

Transaction Hash: 0x8257358907638e4bea2b53b1ae572b0bee2f9e920a59c84b4097e5f31ddf2314

Status: Success

Block: 10716410 18 Block Confirmations

Timestamp: 4 mins ago (May-21-2022 07:44:13 PM +UTC)

From: 0x7cb51c6b92b7d0b5a55cbdecace00f2f6b02f29

To: [Contract 0xc55fccc0cd54dfb6ae1a3766e5bc18759e8ce175 Created]

Value: 0 Ether (\$0.00)

Transaction Fee: 0.000471862603860694 Ether (\$0.00)

Gas Price: 0.000000001100000009 Ether (1.100000009 Gwei)

[Click to see More](#)

A transaction is a cryptographically signed instruction from an account that changes the state of the blockchain. Block explorers track the details of all transactions in the network. Learn more about transactions in our [Knowledge Base](#).

Figure 11 – Etherscan transaction of the acceptance

Etherscan Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Transaction Details < >

Overview State

[This is a Rinkeby Testnet transaction only]

Transaction Hash: 0x8257358907638e4bea2b53b1ae572b0bee2f9e920a59c84b4097e5f31ddf2314

Status: Success

Block: 10716410 18 Block Confirmations

Timestamp: 4 mins ago (May-21-2022 07:44:13 PM +UTC)

From: 0x7cb51c6b92b7d0b5a55cbdecace00f2f6b02f29

To: [Contract 0xc55fccc0cd54dfb6ae1a3766e5bc18759e8ce175 Created]

Value: 0 Ether (\$0.00)

Transaction Fee: 0.000471862603860694 Ether (\$0.00)

Gas Price: 0.000000001100000009 Ether (1.100000009 Gwei)

Gas Limit & Usage by Txn: 536,207 | 428,966 (80%)

Gas Fees: Base: 0.000000008 Gwei

Others: Txn Type: 0 (Legacy) Nonce: 0 Position: 24

Input Data:

```
0x60806040526040518060a0160405280607481526020016106076074913960009080519060200190610032929190610045565b5034801561
003f57600080fd5b50610148565b82805461005190610117565b90600052602060002090601f01602090048101928261007357600085556100
3111156100b957825182
559160200191906001019061009e565b5b5090506100c791906100cb565b5090565b5b808211156100e45760008160009055506001016100cc
```

Figure 11 – Etherscan transaction for viewing the notary

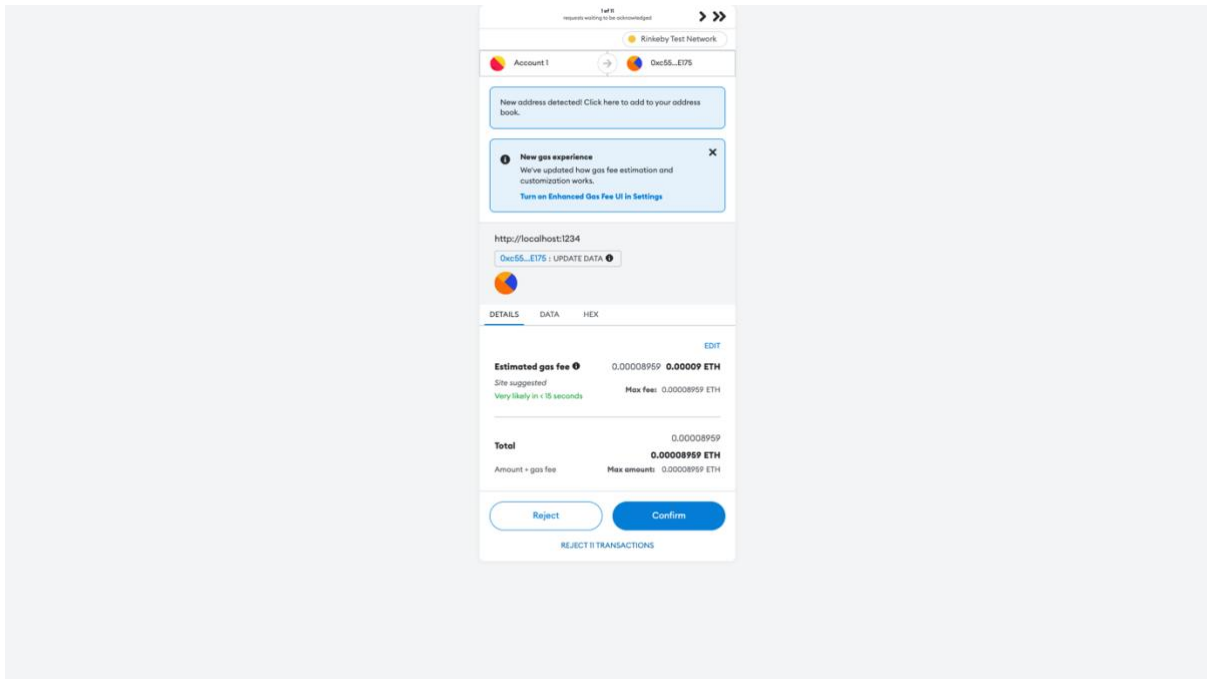


Figure 12 – Integration of the Metamask with localhost

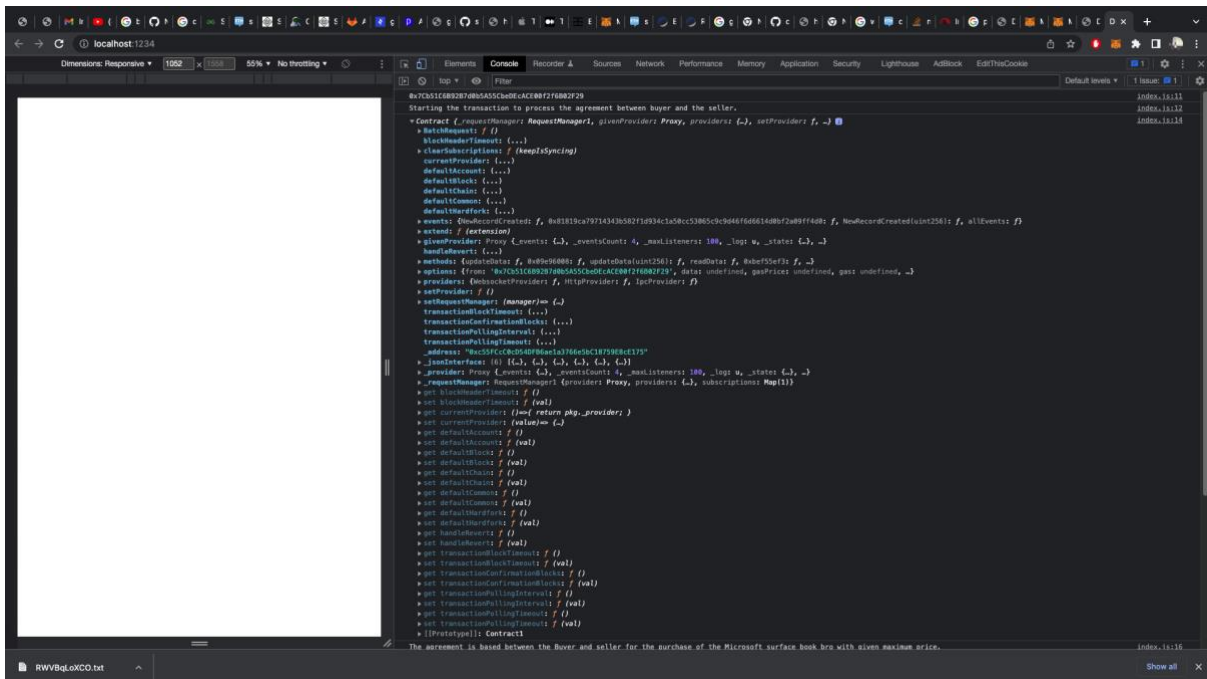


Figure 13 – DAPP running in the browser using Parcel

[illegible]

Figure 14 – DAPP running in the browser using Parcel

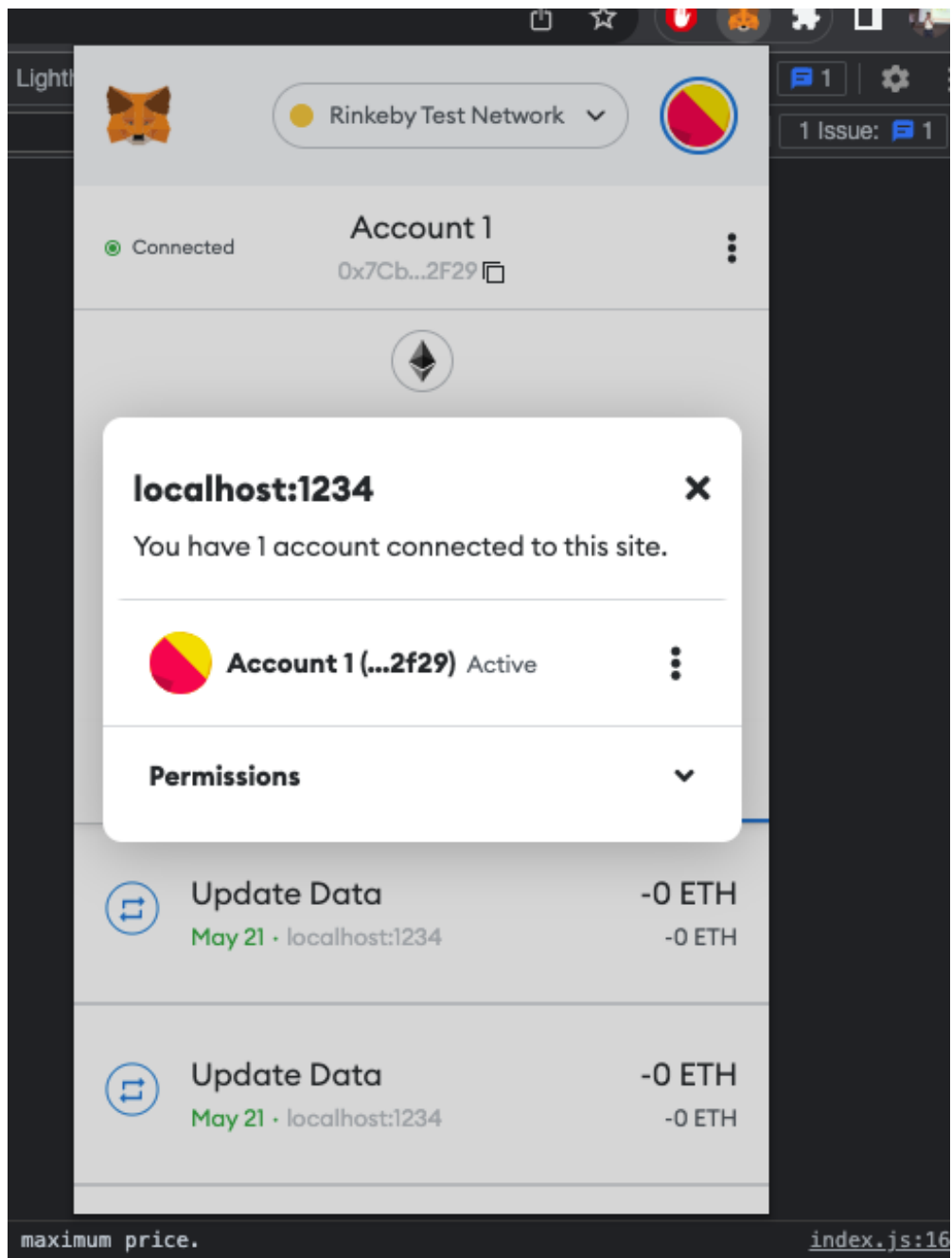


Figure 15 – Connecting Metamask with Transaction to test


Update Data

Status

Confirmed


[View on block explorer](#)
[Copy Transaction ID](#)

From

 0x7Cb...2F29

→

To

 0xc55...E175

Transaction

Nonce	14
Amount	-0 ETH
Gas Limit (Units)	35835
Gas Used (Units)	23802
Base Fee (GWEI)	0.000000008
Priority Fee (GWEI)	2.5
Total Gas Fee	0.00006 ETH
Max Fee Per Gas	0.000000003 ETH
Total	0.00005951 ETH

Figure 16 – Transaction running along with the processed gas

[illegible]

Figure 17 – Transaction running in the browser along with the status

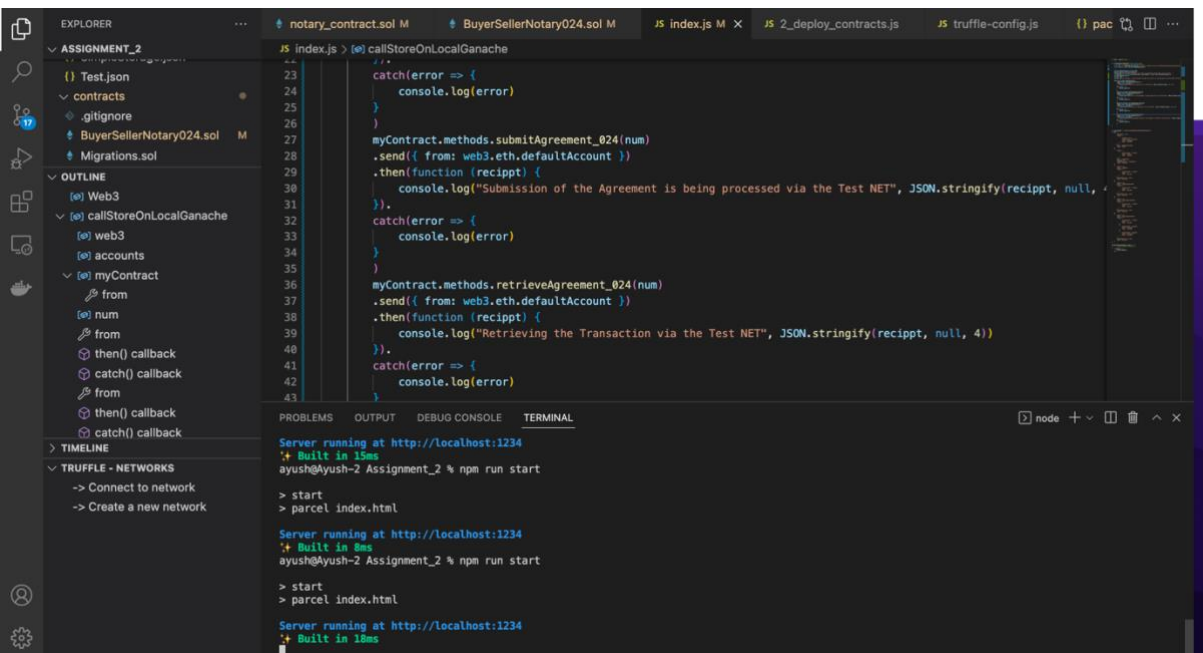


Figure 18 – Parcel running the server using the Node backend

```

    then(function (recippt) {
      console.log("Starting the transaction for the Notary", JSON.stringify(recippt, null, 4))
    }).
    catch(error => {
      console.log(error)
    })
  )
  myContract.methods.submitAgreement_024(num)
    .send({ from: web3.eth.defaultAccount })
    .then(function (recippt) {
      console.log("Submission of the Agreement is being processed via the Test NET", JSON.stringify(recippt, null, 4))
    }).
    catch(error => {
      console.log(error)
    })
  )
  myContract.methods.retrieveAgreement_024(num)
    .send({ from: web3.eth.defaultAccount })
    .then(function (recippt) {
      console.log("Retrieving the Transaction via the Test NET", JSON.stringify(recippt, null, 4))
    }).
    catch(error => {
      console.log(error)
    })
  )
  myContract.methods.approveAgreement_024(num)
    .send({ from: web3.eth.defaultAccount })
    .then(function (recippt) {
      console.log("Approving agreement based on the response of the Buyer and the Seller", JSON.stringify(recippt, null, 4))
    }).
    catch(error => {
      console.log(error)
    })
  )
}

```

Figure 19 – Using Web3 to use the smart contract

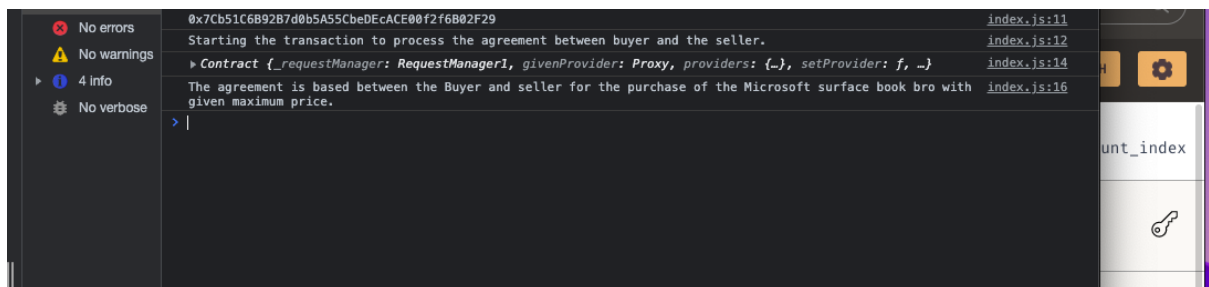


Figure 20 – Running transaction based on the input

References

“The crypto wallet for defi, Web3 Dapps and nfts,” *MetaMask*. [Online]. Available: <https://metamask.io/>. [Accessed: 21-May-2022].

“Ethereum API: Ipfs API & gateway: ETH Nodes as a service,” *Infura*. [Online]. Available: <https://infura.io/>. [Accessed: 21-May-2022].

“Ethereum Development Documentation,” *ethereum.org*. [Online]. Available: <https://ethereum.org/en/developers/docs/>. [Accessed: 21-May-2022].

“Log in to access the Lucid Visual Collaboration Suite,” *Lucid visual collaboration suite: Log in*. [Online]. Available: <https://lucid.app/users/login>. [Accessed: 21-May-2022].