DALHOUSIE UNIVERSITY

CSCI 6313 – Introduction to Blockchains

Assignment Number
2

Student Name
Aman Singh Bhandari

Banner ID
B00910008

Gitlab URL
https://git.cs.dal.ca/bhandari/csci-6313-assignment2

Table of Contents

Gitlab Repository URL

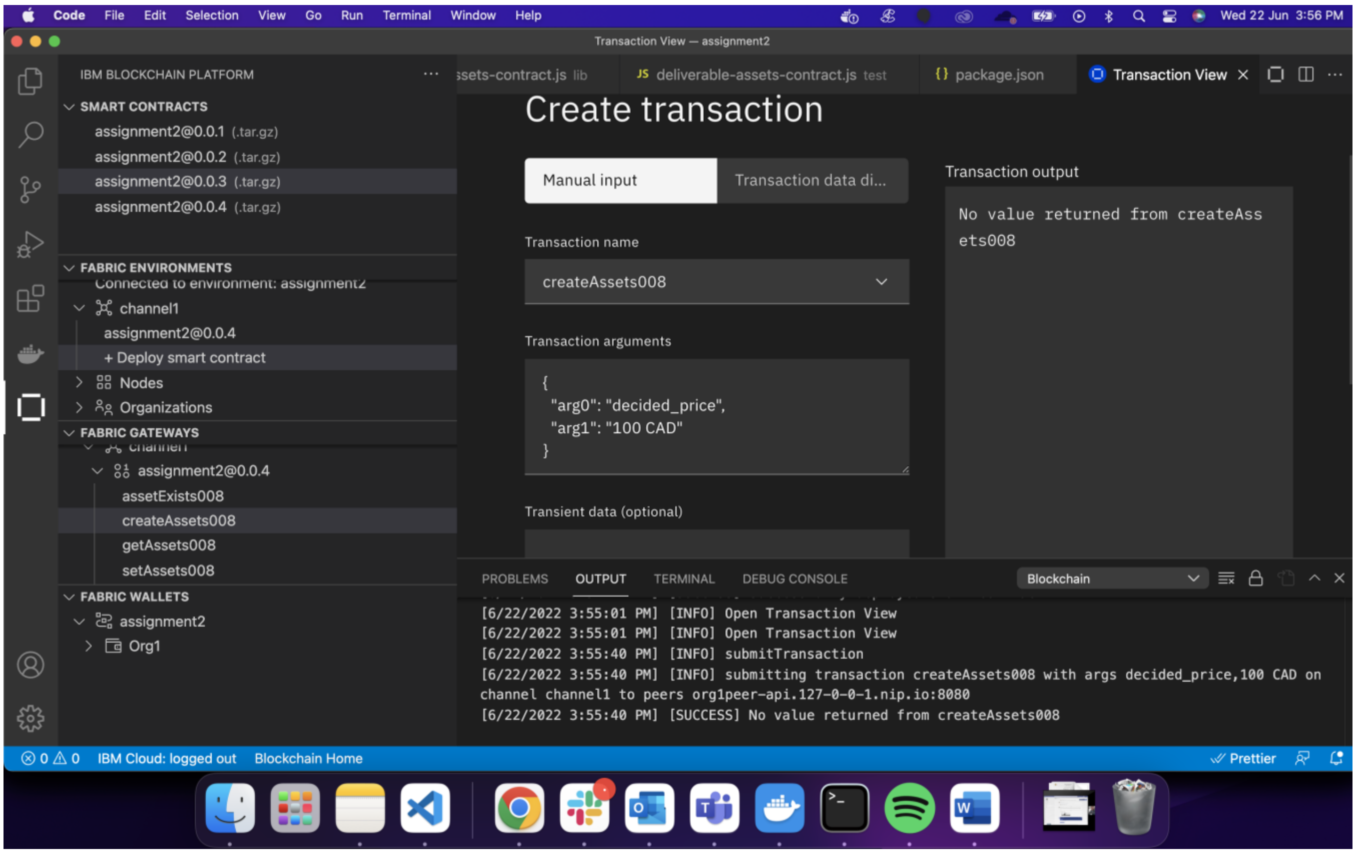
https://git.cs.dal.ca/bhandari/csci-6313-assignment2

This assignment is done in JavaScript using Visual Studio.

Part A

Written a new smart contract class for this part of the assignment -> Changed the version in package.json -> packaged with tar.gz -> Deploy through IDE -> Tested through IDE.

Create State



Create transaction

Transaction name: createAssets008

Transaction arguments

```
{
  "arg0": "decided_price",
  "arg1": "100 CAD"
}
```

Transient data (optional)

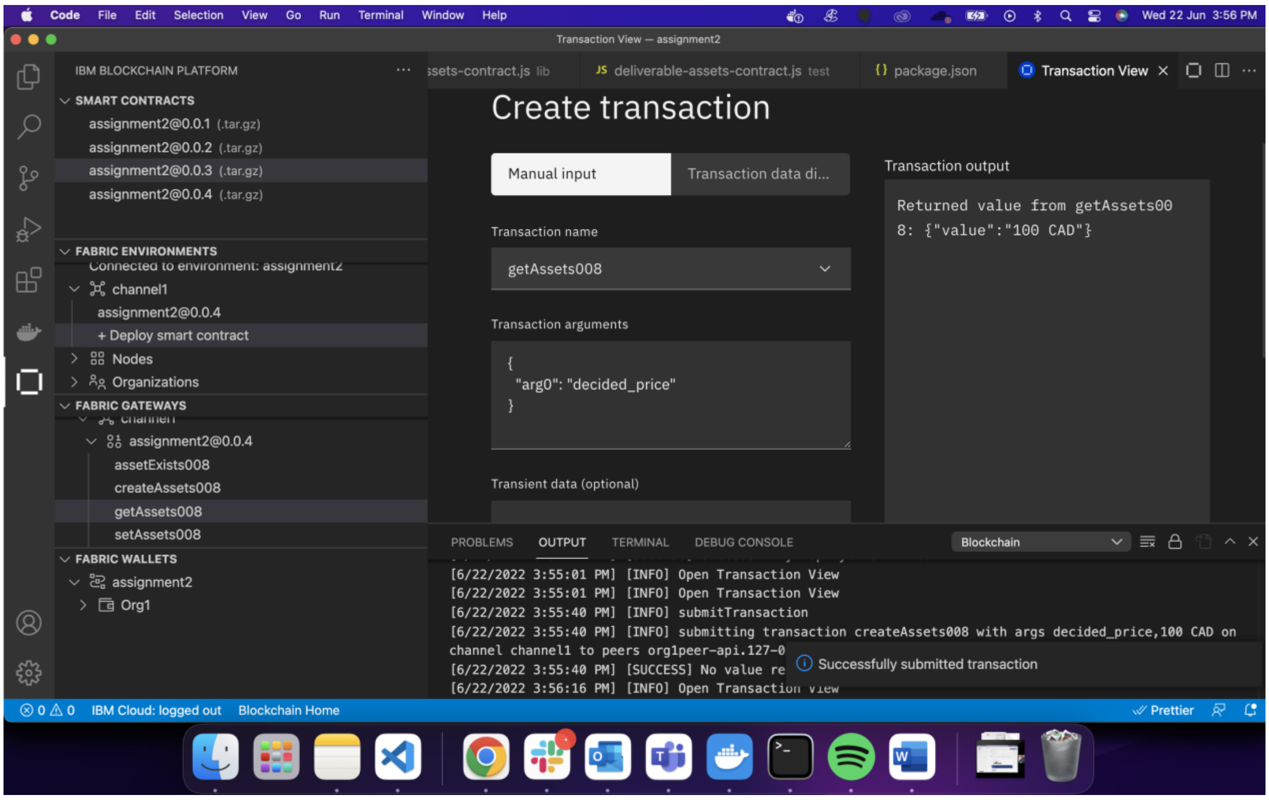
Transaction output

```
No value returned from createAssets008
```

```
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:40 PM] [INFO] submitTransaction
[6/22/2022 3:55:40 PM] [INFO] submitting transaction createAssets008 with args decided_price,100 CAD on
channel channel1 to peers org1peer-api.127-0-0-1.nip.io:8080
[6/22/2022 3:55:40 PM] [SUCCESS] No value returned from createAssets008
```

Figure 1 Create state method

Get State



Create transaction

Transaction name: getAssets008

Transaction arguments

```
{
  "arg0": "decided_price"
}
```

Transient data (optional)

Transaction output

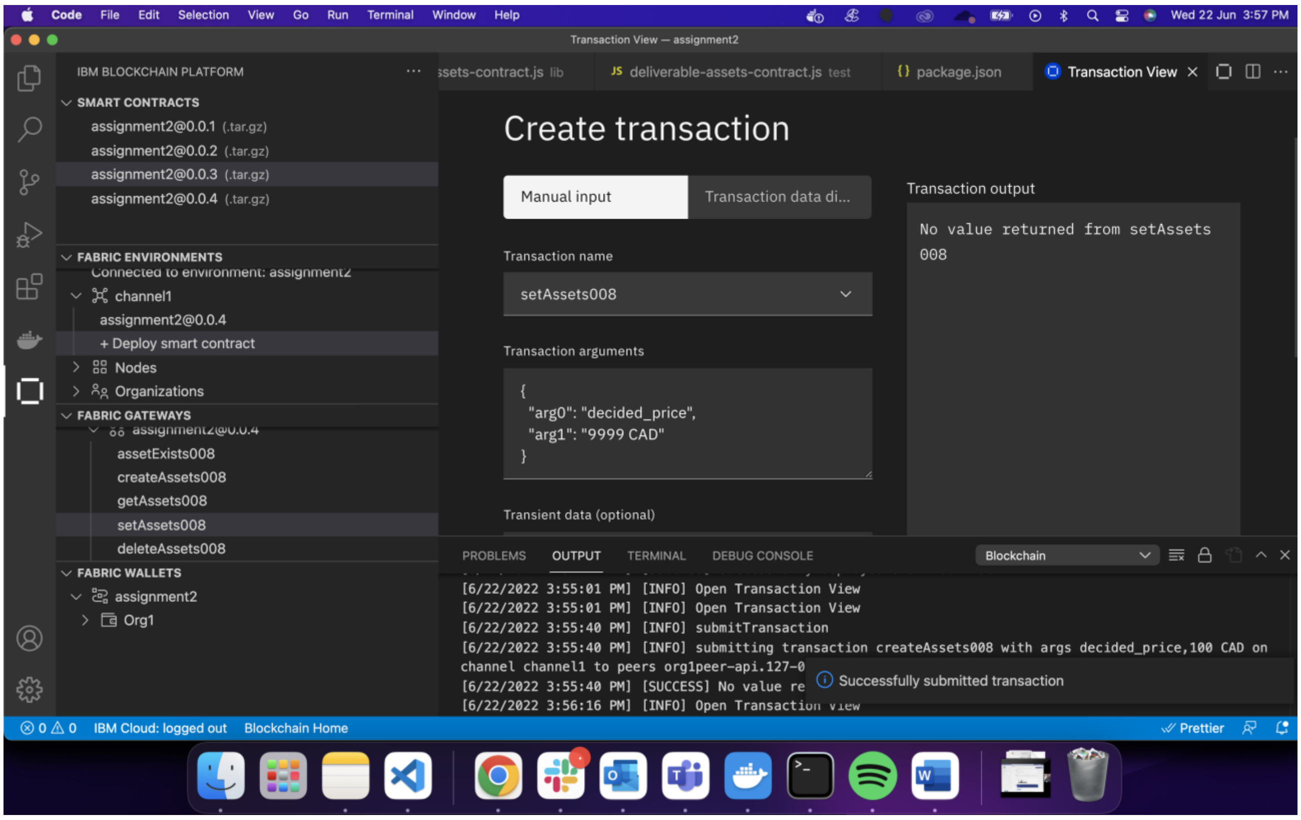
```
Returned value from getAssets00
8: {"value":"100 CAD"}
```

```
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:40 PM] [INFO] submitTransaction
[6/22/2022 3:55:40 PM] [INFO] submitting transaction createAssets008 with args decided_price,100 CAD on
channel channel1 to peers org1peer-api.127-0
[6/22/2022 3:55:40 PM] [SUCCESS] No value re
[6/22/2022 3:56:16 PM] [INFO] Open Transaction View
```

Successfully submitted transaction

Figure 2 read method

Set State



```
{
  "arg0": "decided_price",
  "arg1": "9999 CAD"
}
```

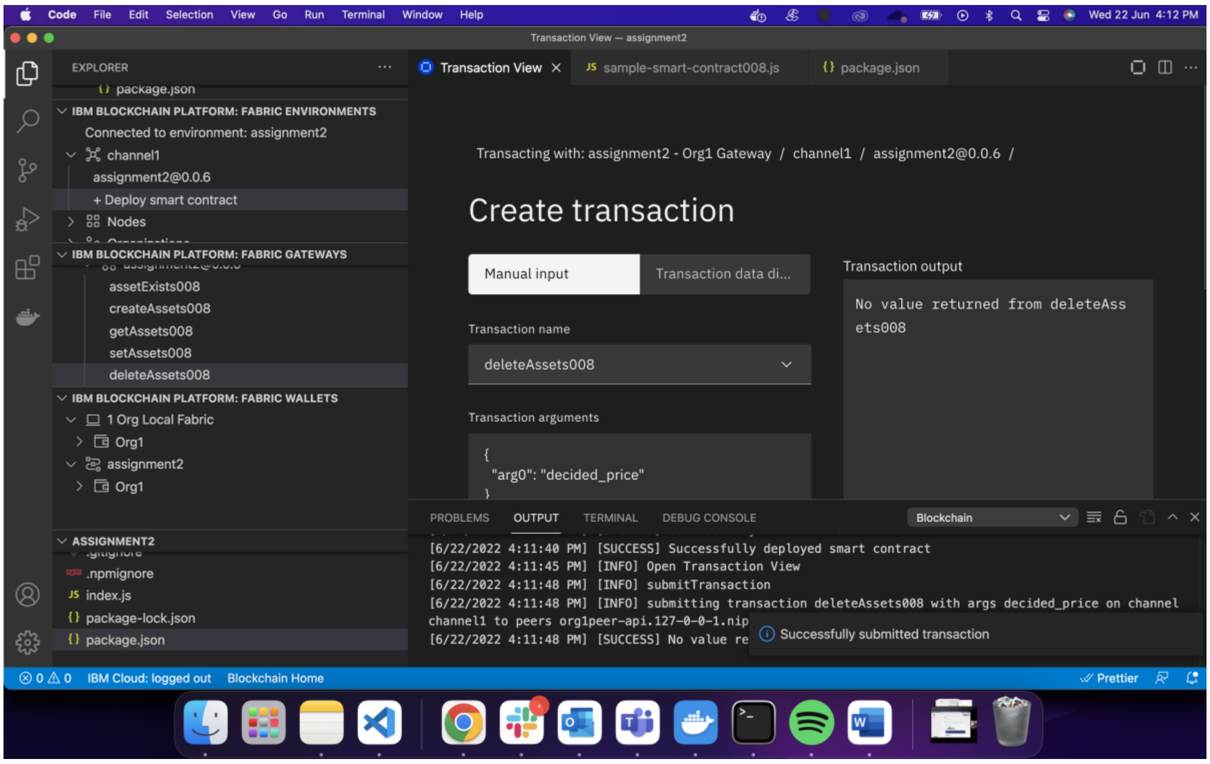
```
No value returned from setAssets
008
```

```
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:01 PM] [INFO] Open Transaction View
[6/22/2022 3:55:40 PM] [INFO] submitTransaction
[6/22/2022 3:55:40 PM] [INFO] submitting transaction createAssets008 with args decided_price,100 CAD on
channel channel1 to peers org1peer-api.127-0
[6/22/2022 3:55:40 PM] [SUCCESS] No value re
[6/22/2022 3:56:16 PM] [INFO] Open Transaction View
```

Successfully submitted transaction

Figure 3 update method

Delete State



Transacting with: assignment2 - Org1 Gateway / channel1 / assignment2@0.0.6 /

Create transaction

Transaction name: deleteAssets008

Transaction arguments

```
{
 "arg0": "decided_price"
}
```

Transaction output

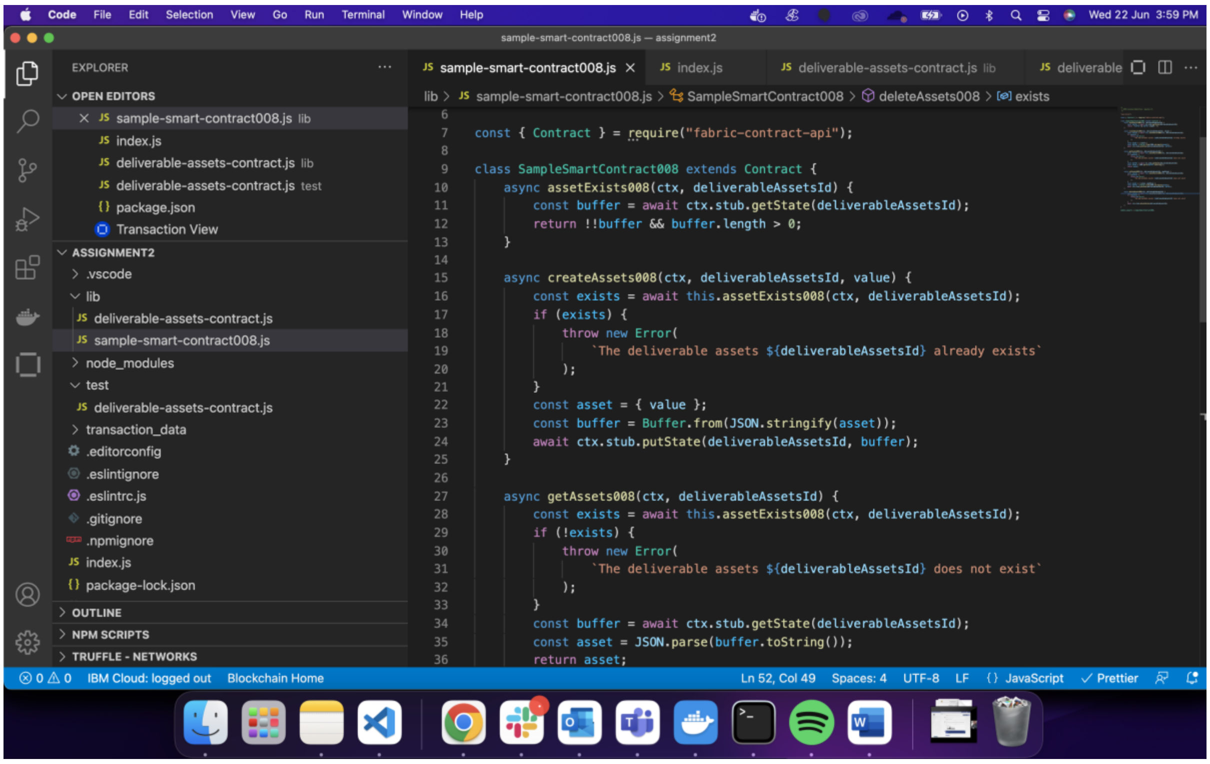
```
No value returned from deleteAssets008
```

```
[6/22/2022 4:11:40 PM] [SUCCESS] Successfully deployed smart contract
[6/22/2022 4:11:45 PM] [INFO] Open Transaction View
[6/22/2022 4:11:48 PM] [INFO] submitTransaction
[6/22/2022 4:11:48 PM] [INFO] submitting transaction deleteAssets008 with args decided_price on channel
channel1 to peers org1peer-api.127-0-0-1.nip
[6/22/2022 4:11:48 PM] [SUCCESS] No value re
```

Successfully submitted transaction

Figure 4 delete method

Smart Contract Class



```
const { Contract } = require("fabric-contract-api");

class SampleSmartContract008 extends Contract {
    async assetExists008(ctx, deliverableAssetsId) {
        const buffer = await ctx.stub.getState(deliverableAssetsId);
        return !!buffer && buffer.length > 0;
    }

    async createAssets008(ctx, deliverableAssetsId, value) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (exists) {
            throw new Error(
                `The deliverable assets ${deliverableAssetsId} already exists`
            );
        }
        const asset = { value };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(deliverableAssetsId, buffer);
    }

    async getAssets008(ctx, deliverableAssetsId) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
                `The deliverable assets ${deliverableAssetsId} does not exist`
            );
        }
        const buffer = await ctx.stub.getState(deliverableAssetsId);
        const asset = JSON.parse(buffer.toString());
        return asset;
```

Code File Edit Selection View Go Run Terminal Window Help
Wed 22 Jun 3:59 PM
sample-smart-contract008.js — assignment2
EXPLORER
OPEN EDITORS
sample-smart-contract008.js lib
index.js
deliverable-assets-contract.js lib
deliverable-assets-contract.js test
package.json
Transaction View
ASSIGNMENT2
.vscode
lib
deliverable-assets-contract.js
sample-smart-contract008.js
node_modules
test
deliverable-assets-contract.js
transaction_data
.editorconfig
.eslintignore
.eslintrc.js
.gitignore
.npmignore
index.js
package-lock.json
OUTLINE
NPM SCRIPTS
TRUFFLE - NETWORKS
lib > sample-smart-contract008.js > SampleSmartContract008 > deleteAssets008 > exists
Ln 52, Col 49 Spaces: 4 UTF-8 LF JavaScript Prettier
IBM Cloud: logged out
Blockchain Home

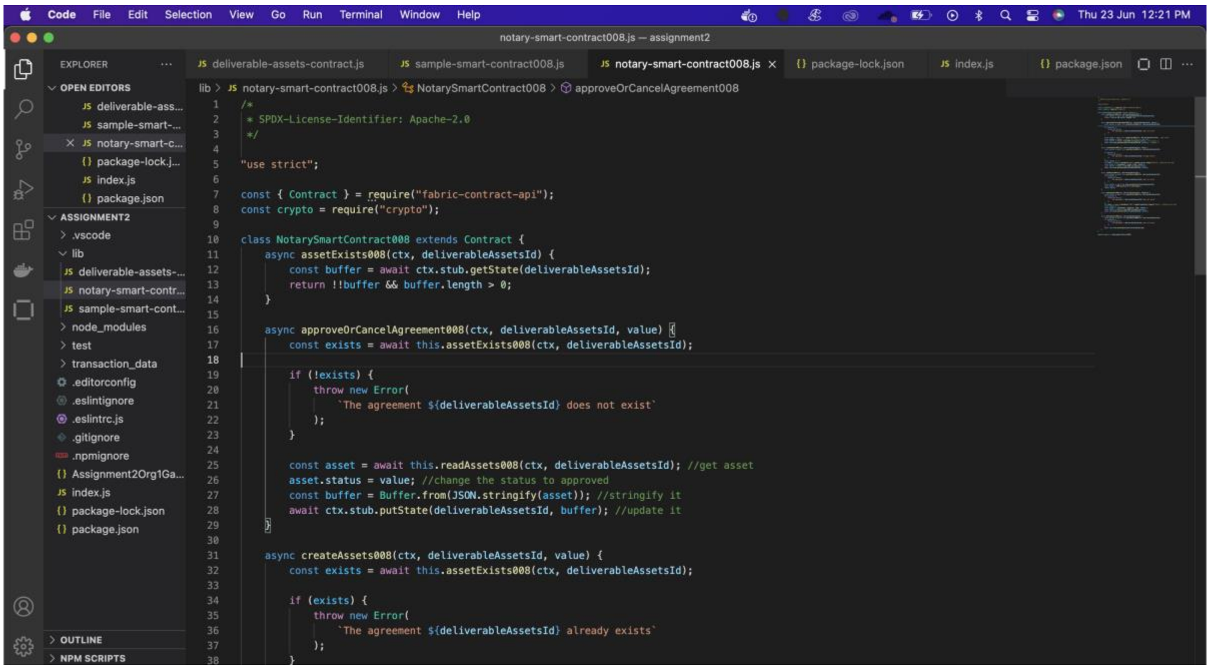
Figure 5 Smart contract class

Part B – Create a Dapp that will act as a Notary for Buyer and Seller (in Hyperledger)

Smart Contract class

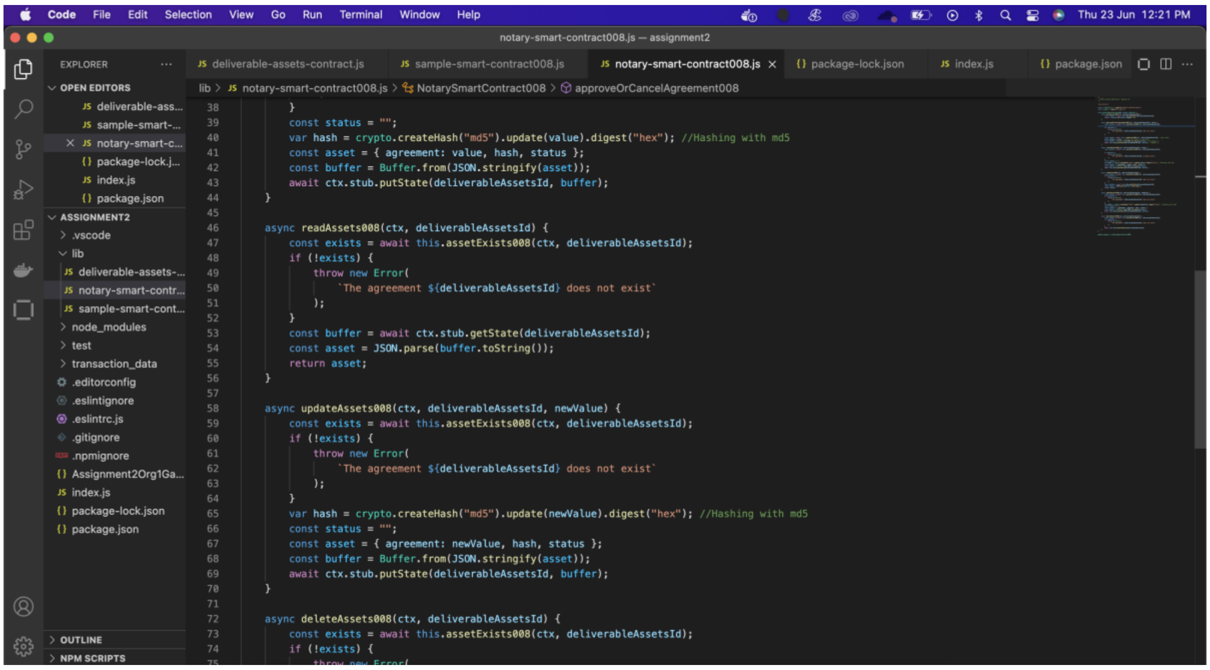
The class is built in a way that it is able to submit an agreement between Buyer and Seller. While submitting the agreement, it calculates the hash (through md5) and save it as well in the state (with key name "hash"). During create, I am also assigning key in the state – Status as empty string. This state will save the agreement status. When the contract is approved, the status will be assigned "approved" or "cancelled" otherwise.

Below are some screenshots of how the smart contract class is designed.



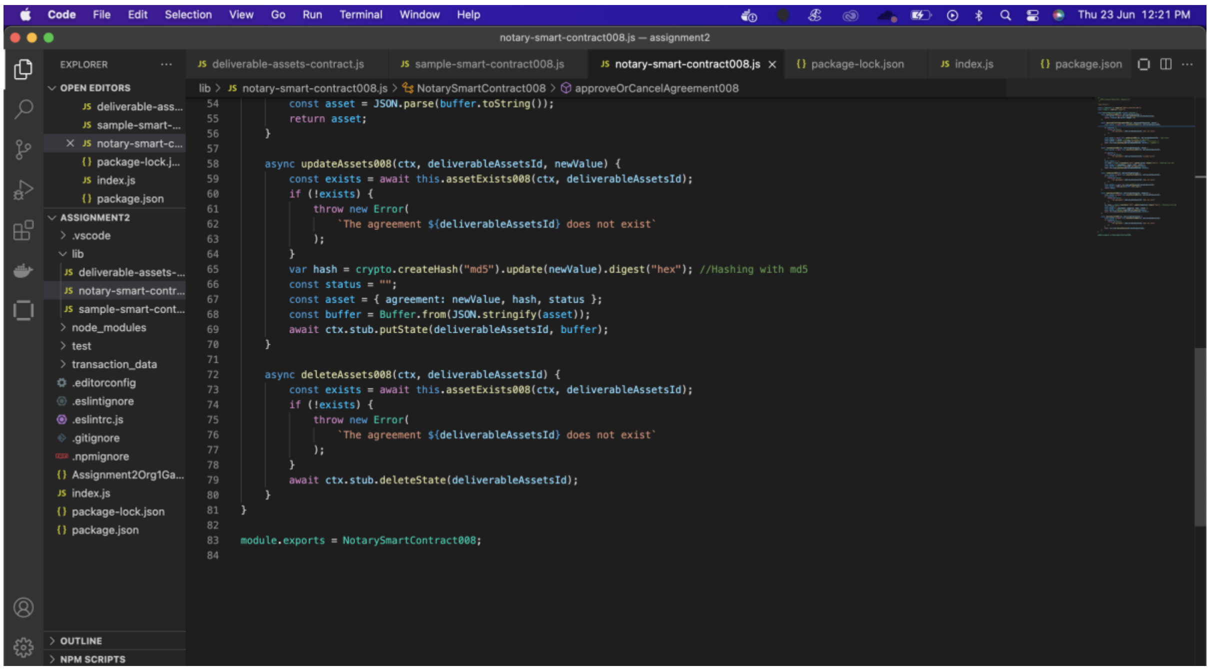
```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

"use strict";

const { Contract } = require("fabric-contract-api");
const crypto = require("crypto");

class NotarySmartContract008 extends Contract {
    async assetExists008(ctx, deliverableAssetsId) {
        const buffer = await ctx.stub.getState(deliverableAssetsId);
        return !!buffer && buffer.length > 0;
    }

    async approveOrCancelAgreement008(ctx, deliverableAssetsId, value) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);

        if (!exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} does not exist`
            );
        }

        const asset = await this.readAssets008(ctx, deliverableAssetsId); //get asset
        asset.status = value; //change the status to approved
        const buffer = Buffer.from(JSON.stringify(asset)); //stringify it
        await ctx.stub.putState(deliverableAssetsId, buffer); //update it
    }

    async createAssets008(ctx, deliverableAssetsId, value) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);

        if (exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} already exists`
            );
        }
```

Figure 6 smart contract class



```
        }
        const status = "";
        var hash = crypto.createHash("md5").update(value).digest("hex"); //Hashing with md5
        const asset = { agreement: value, hash, status };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(deliverableAssetsId, buffer);
    }

    async readAssets008(ctx, deliverableAssetsId) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} does not exist`
            );
        }
        const buffer = await ctx.stub.getState(deliverableAssetsId);
        const asset = JSON.parse(buffer.toString());
        return asset;
    }

    async updateAssets008(ctx, deliverableAssetsId, newValue) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} does not exist`
            );
        }
        var hash = crypto.createHash("md5").update(newValue).digest("hex"); //Hashing with md5
        const status = "";
        const asset = { agreement: newValue, hash, status };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(deliverableAssetsId, buffer);
    }

    async deleteAssets008(ctx, deliverableAssetsId) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
```

Figure 7 smart contract class - cont.



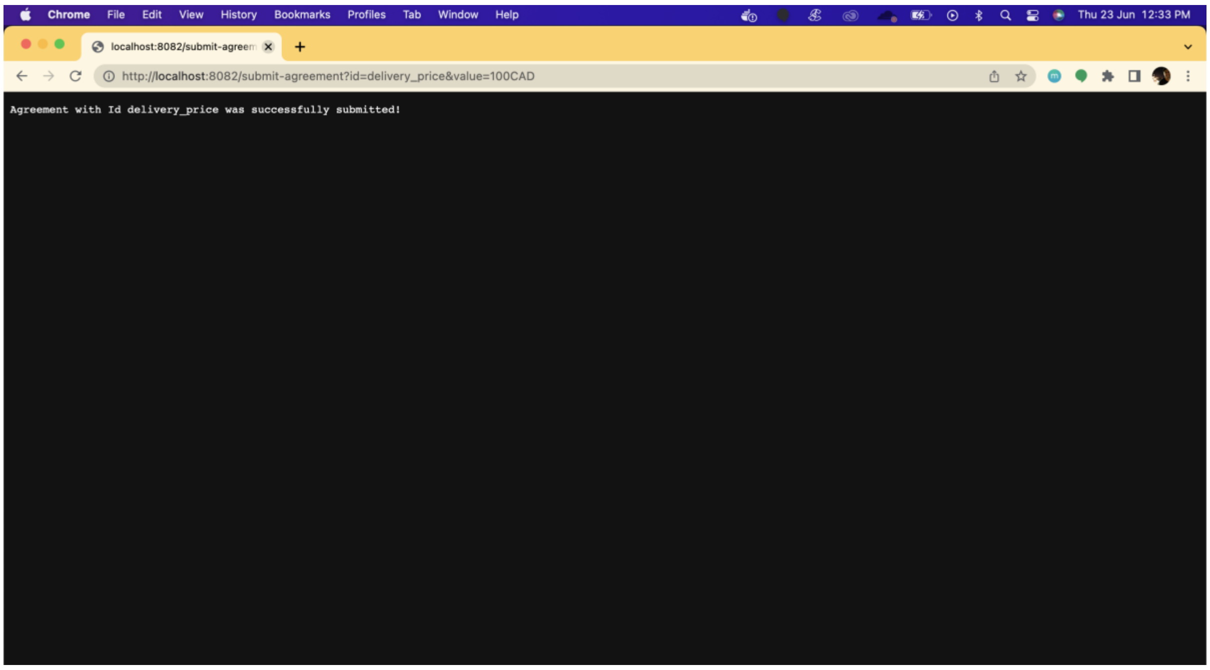
```
        const asset = JSON.parse(buffer.toString());
        return asset;
    }

    async updateAssets008(ctx, deliverableAssetsId, newValue) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} does not exist`
            );
        }
        var hash = crypto.createHash("md5").update(newValue).digest("hex"); //Hashing with md5
        const status = "";
        const asset = { agreement: newValue, hash, status };
        const buffer = Buffer.from(JSON.stringify(asset));
        await ctx.stub.putState(deliverableAssetsId, buffer);
    }

    async deleteAssets008(ctx, deliverableAssetsId) {
        const exists = await this.assetExists008(ctx, deliverableAssetsId);
        if (!exists) {
            throw new Error(
                `The agreement ${deliverableAssetsId} does not exist`
            );
        }
        await ctx.stub.deleteState(deliverableAssetsId);
    }
}

module.exports = NotarySmartContract008;
```

Figure 8 smart contract class (cont..)

Dapp – Submit Agreement

Dapp is written using javascript. For submit agreement it first create the network gateway, use the identity and then gets the contract in the specified channel. Once initialization is done, the javascript app calls create method.

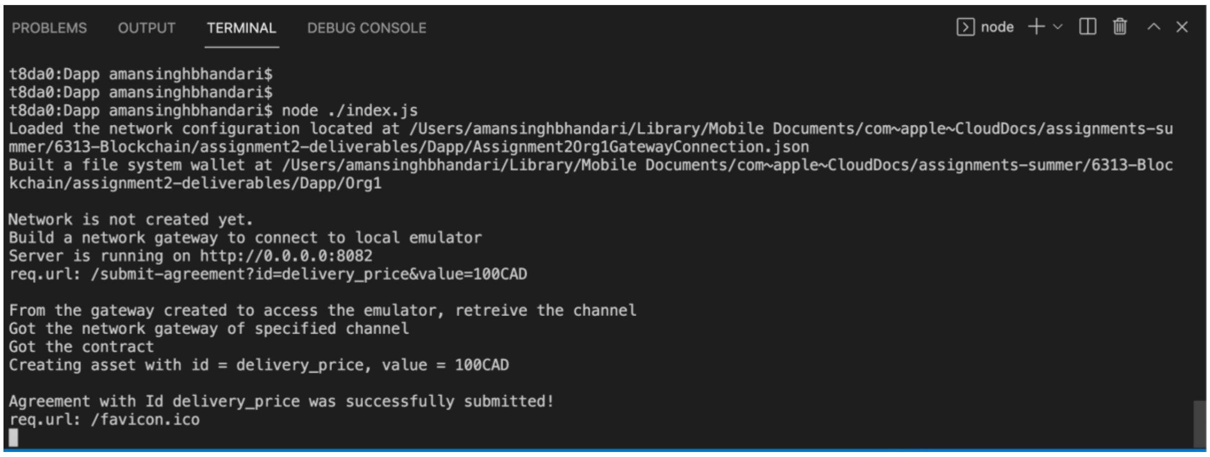
Below is the web api call, output of console log.



Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu 23 Jun 12:33 PM
localhost:8082/submit-agreem
http://localhost:8082/submit-agreement?id=delivery_price&value=100CAD

```
Agreement with Id delivery_price was successfully submitted!
```

Figure 9 submit agreement (create) output

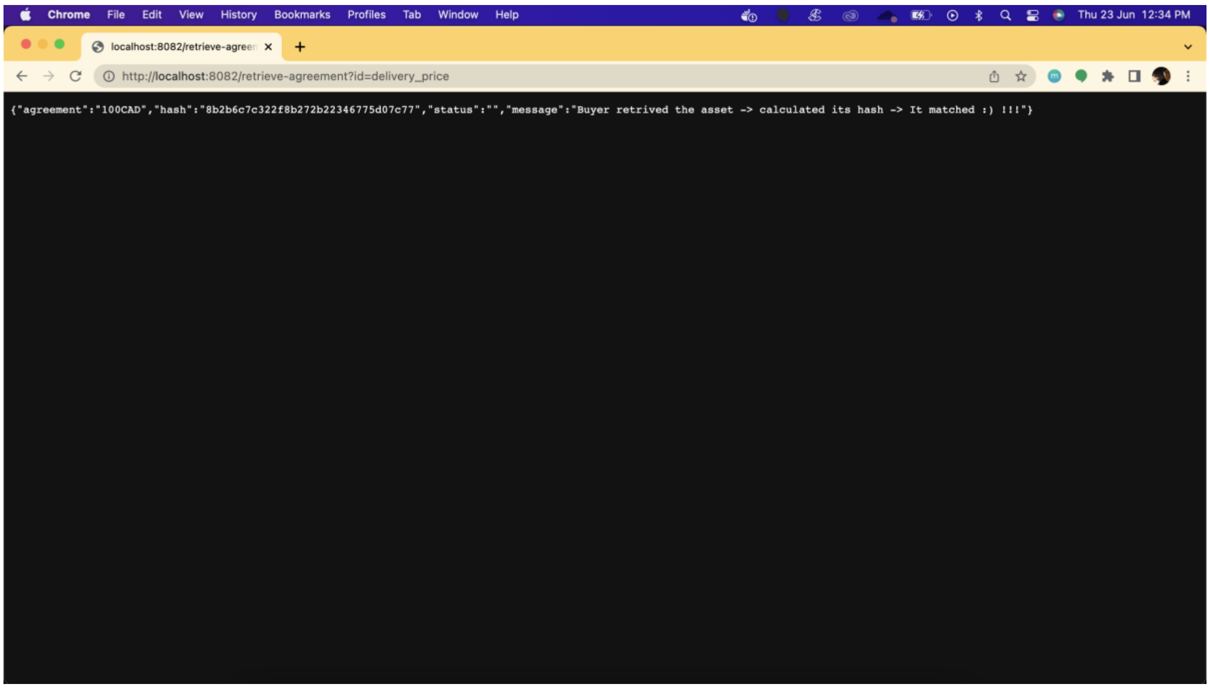


```
t8da0:Dapp amansinghbhandari$
t8da0:Dapp amansinghbhandari$
t8da0:Dapp amansinghbhandari$ node ./index.js
Loaded the network configuration located at /Users/amansinghbhandari/Library/Mobile Documents/com~apple~CloudDocs/assignments-su
mmer/6313-Blockchain/assignment2-deliverables/Dapp/Assignment2Org1GatewayConnection.json
Built a file system wallet at /Users/amansinghbhandari/Library/Mobile Documents/com~apple~CloudDocs/assignments-summer/6313-Bloc
kchain/assignment2-deliverables/Dapp/Org1

Network is not created yet.
Build a network gateway to connect to local emulator
Server is running on http://0.0.0.0:8082
req.url: /submit-agreement?id=delivery_price&value=100CAD

From the gateway created to access the emulator, retreive the channel
Got the network gateway of specified channel
Got the contract
Creating asset with id = delivery_price, value = 100CAD

Agreement with Id delivery_price was successfully submitted!
req.url: /favicon.ico
```

Figure 10 submit agreement output (console log)

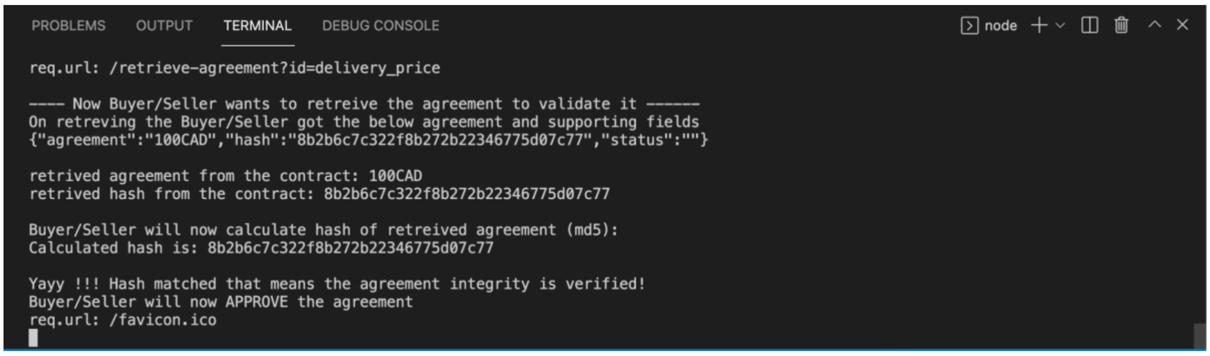
Dapp – Retrieve Agreement

In this web API, I am retrieving the agreement and its hash from the deployed contract. Once it is received, generating the hash of the agreement again in Dapp and then comparing it with the one received. You can check the same in the console logs. The “message” in the json response is not returned from the contract, instead it was added in Dapp just to show on the browser.


localhost:8082/retrieve-agreen
http://localhost:8082/retrieve-agreement?id=delivery_price

```
{"agreement":"100CAD","hash":"8b2b6c7c322f8b272b22346775d07c77","status":"","message":"Buyer retrived the asset -> calculated its hash -> It matched :) !!!"}
```

Figure 11 retrieve agreement (read) output

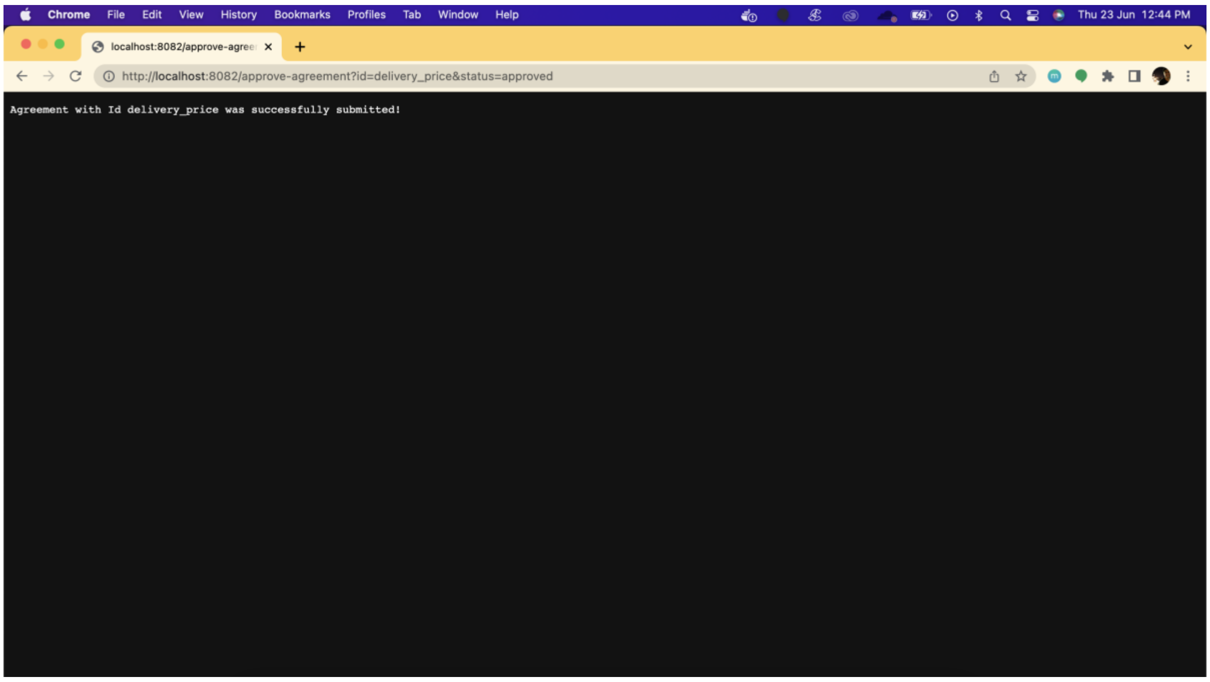


```
req.url: /retrieve-agreement?id=delivery_price

---- Now Buyer/Seller wants to retreive the agreement to validate it ------
On retreving the Buyer/Seller got the below agreement and supporting fields
{"agreement":"100CAD","hash":"8b2b6c7c322f8b272b22346775d07c77","status":""}

retrived agreement from the contract: 100CAD
retrived hash from the contract: 8b2b6c7c322f8b272b22346775d07c77

Buyer/Seller will now calculate hash of retreived agreement (md5):
Calculated hash is: 8b2b6c7c322f8b272b22346775d07c77

Yayy !!! Hash matched that means the agreement integrity is verified!
Buyer/Seller will now APPROVE the agreement
req.url: /favicon.ico
```

Figure 12 retrieve agreement output (console log)

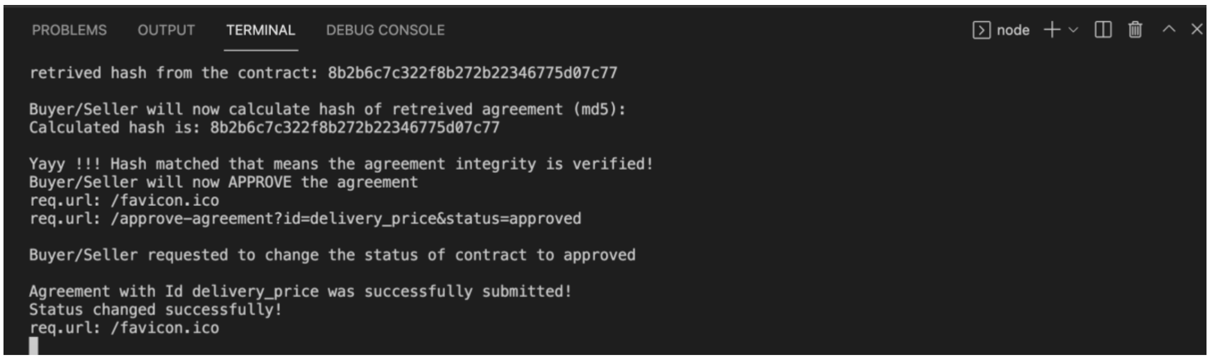
Dapp – Approve or Cancel the agreement

On retrieval buyer/seller is checking whether the hash is matched or not. Upon this, buyer/seller will take the decision to approve or cancel the contract. Let's call our next api to approve or cancel agreement. As you can see in the figure below, I am passing the approved or cancelled in the parameter "status".


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu 23 Jun 12:44 PM
localhost:8082/approve-agree
http://localhost:8082/approve-agreement?id=delivery_price&status=approved

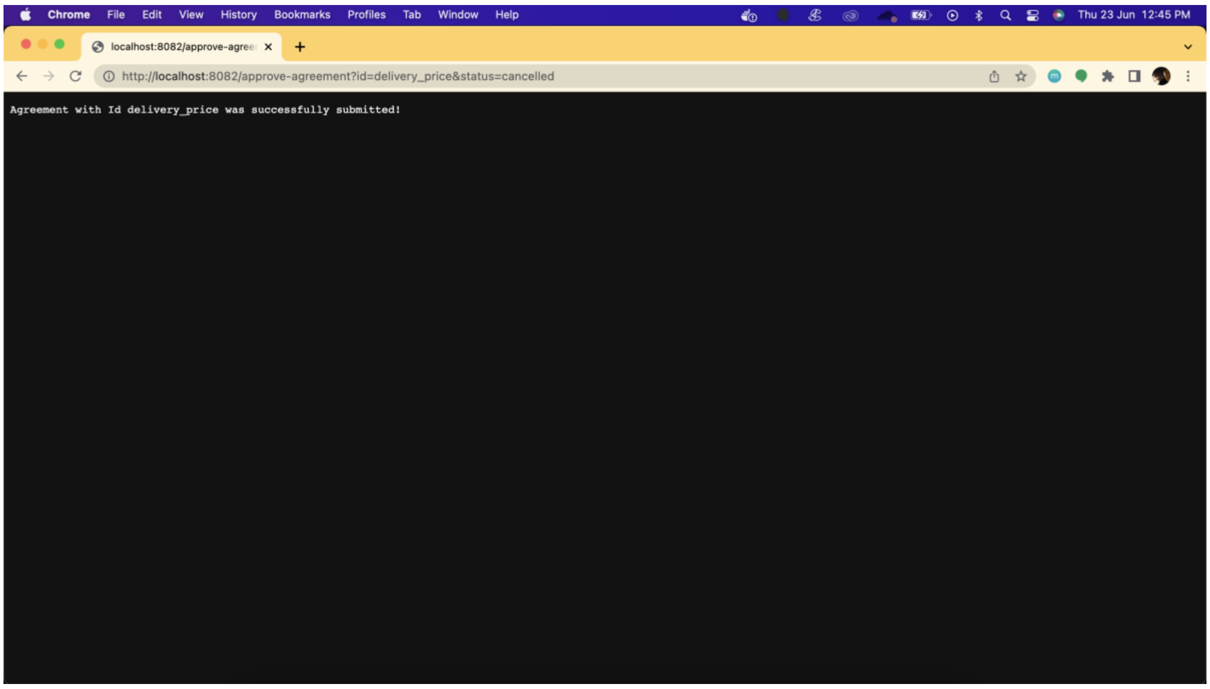
```
Agreement with Id delivery_price was successfully submitted!
```

Figure 13 approve agreement (update) output



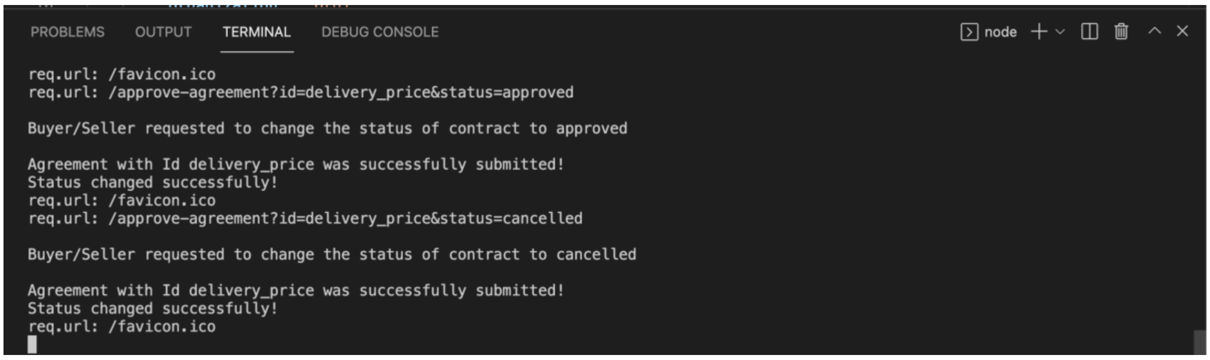
```
retrived hash from the contract: 8b2b6c7c322f8b272b22346775d07c77

Buyer/Seller will now calculate hash of retreived agreement (md5):
Calculated hash is: 8b2b6c7c322f8b272b22346775d07c77

Yayy !!! Hash matched that means the agreement integrity is verified!
Buyer/Seller will now APPROVE the agreement
req.url: /favicon.ico
req.url: /approve-agreement?id=delivery_price&status=approved

Buyer/Seller requested to change the status of contract to approved

Agreement with Id delivery_price was successfully submitted!
Status changed successfully!
req.url: /favicon.ico
```

Figure 14 approve agreement output (console log)


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu 23 Jun 12:45 PM
localhost:8082/approve-agree
http://localhost:8082/approve-agreement?id=delivery_price&status=cancelled

```
Agreement with Id delivery_price was successfully submitted!
```

Figure 15 cancel agreement (update) output



```
req.url: /favicon.ico
req.url: /approve-agreement?id=delivery_price&status=approved

Buyer/Seller requested to change the status of contract to approved

Agreement with Id delivery_price was successfully submitted!
Status changed successfully!
req.url: /favicon.ico
req.url: /approve-agreement?id=delivery_price&status=cancelled

Buyer/Seller requested to change the status of contract to cancelled

Agreement with Id delivery_price was successfully submitted!
Status changed successfully!
req.url: /favicon.ico
```

Figure 16 cancel agreement output (console log)