

Technial Documentation

LAB 3 : Supply Chain Project

By : Fajri Nurfauzan | 1103180184

Pada lab kedua untuk UTS ini akan mempelajari cara sebuah Manager yang dapat menambah, membayar, dan memindahkan sebuah item atau wallet ke dalam *Supply Chain* yang akan memicu pengiriman melewati solusi rantai dengan contoh kasus seperti :

- Dapat menjadi bagian dari solusi rantai pasokan
- Pengiriman Otomatis setelah pembayaran
- Penagihan pembayaran tanpa perantara

Setelah memahami teori penggunaan dan contoh studi kasus bisa langsung mengikuti langkah berikut :

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "./Item.sol";
6
7 contract ItemManager {
8
9     struct S_Item {
10         Item _item;
11         ItemManager.SupplyChainSteps _step;
12         string _identifier;
13     }
14
15     mapping(uint => S_Item) public items;
16     uint index;
17
18     enum SupplyChainSteps {Created, Paid, Delivered}
19     event SupplyChainStep(uint _itemIndex, uint _step, address _address);
20
21     function createItem(string memory _identifier, uint _priceInWei) public {
22         Item item = new Item(this, _priceInWei, index);
23         items[index]._item = item;
24         items[index]._step = SupplyChainSteps.Created;
25         items[index]._identifier = _identifier;
26         emit SupplyChainStep(index, uint(items[index]._step), address(item));
27         index++;
28     }
29
30     function triggerPayment(uint _index) public payable {
31         Item item = items[_index]._item;
32         require(address(item) == msg.sender, "Only items are allowed to update themselves");
33         require(item.priceInWei() == msg.value, "Not fully paid yet");
34         require(items[_index]._step == SupplyChainSteps.Created, "Item is further in the supply chain");
35         items[_index]._step = SupplyChainSteps.Paid;
36         emit SupplyChainStep(_index, uint(items[_index]._step), address(item));
37     }
38
39     function triggerDelivery(uint _index) public {
40         require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in the supply chain");
41         items[_index]._step = SupplyChainSteps.Delivered;
42         emit SupplyChainStep(_index, uint(items[_index]._step), address(items[_index]._item));
43     }
44 }
```

Pertama kita membuat code solidity tersebut boleh di remix boleh melalui vscode dengan nama “ItemManager.sol” dikarenakan code tersebut akan digunakan untuk langkah selanjutnya ketika menginstall beberapa aplikasi, setelah menulis code diatas lakukan penulisan kembali untuk menulis code berikut dan beri nama dengan “Item.sol”

```

1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "./ItemManager.sol";
6
7 contract Item {
8     uint public priceInWei;
9     uint public paidWei;
10    uint public index;
11
12    ItemManager parentContract;
13    constructor(ItemManager _parentContract, uint _priceInWei, uint _index) public {
14        priceInWei = _priceInWei;
15        index = _index;
16        parentContract = _parentContract;
17    }
18
19    receive() external payable {
20        require(msg.value == priceInWei, "We don't support partial payments");
21        require(paidWei == 0, "Item is already paid!");
22        paidWei += msg.value;
23        (bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));
24        require(success, "Delivery did not work");
25    }
26
27    fallback () external {
28
29    }
30 }

```

Sekarang dengan ini, hanya perlu memberi nama pelanggan dengan alamat Kontrak Cerdas (*Smart Contract*) Item yang dibuat dengan nama "createItem" dan dia akan dapat membayar langsung dengan mengirimkan X Wei ke Kontrak Cerdas (*Smart Kontrak*). Tetapi kontrak pintar (*Smart Kontrak*) yang sudah dibuat belum terlalu aman. Maka langkah selanjutnya butuh semacam fungsi pemilik, dan biasanya code akan ditambahkan Kontrak Cerdas (*Smart Kontrak*) OpenZeppelin dengan Fungsionalitas yang Dapat Dimiliki. Tetapi dikarenakan dokumen mereka belum diperbarui ke solidity 0.6 maka fungsi *Ownable* dibuat sendiri yang sangat mirip dengan satu dari OpenZeppelin dan beri nama file tersebut dengan "Ownable.sol" lalu Dengan ubah ItemManager sehingga semua fungsi, yang seharusnya dapat dieksekusi oleh "pemilik saja" memiliki pengubah yang benar :

```

1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "./Ownable.sol";
6 import "./Item.sol";
7
8 contract ItemManager is Ownable {
9
10    //...
11
12    function createItem(string memory _identifier, uint _priceInWei) public onlyOwner {
13        //...
14    }
15
16    function triggerPayment(uint _index) public payable {
17        //...
18    }
19
20    function triggerDelivery(uint _index) public onlyOwner {
21        //...
22    }
23 }

```

Setelah menulis semua code berikut lakukan install Truffle dengan membuka terminal (Mac/Linux) atau PowerShell (Windows 10/11)

```

PS C:\Users\Fajri> npm install -g truffle
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please switch to that.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated ipfs-http-client@0.10.0: This module has been superseded by the multiformats module
npm WARN deprecated circular-json@0.5.9: CircularJSON is in maintenance only, flattened is its successor.
npm WARN deprecated cids@1.1.9: This module has been superseded by the multiformats module
npm WARN deprecated ipfs-dag-cbor@0.17.1: This module has been superseded by @ipld/dag-cbor and multiformats
npm WARN deprecated multicodec@1.0.4: This module has been superseded by the multiformats module
npm WARN deprecated @nodefactory/filspan-adapter@0.2.2: Package is deprecated in favour of @chainsafe/filspan-adapter

```

Jika sudah bisa cek truffle version dengan cara menulis code berikut :

```
PS C:\Users\Fajri> truffle version
Truffle v5.4.0 (core: 5.4.0)
Solidity v0.5.16 (solc-js)
Node v16.13.0
Web3.js v1.4.0
```

Jika sudah silahkan membuat Kemudian buat folder kosong, dengan nama "s06-eventtrigger" dan jika sudah bisa menggunakan fungsi cd s06-eventtrigger untuk berpindah folder

```
ebd> mkdir s06-eventtrigger
```

Dan mengetikkan ls untuk melihat isi folder

```
ebd> cd .\s06-eventtrigger\
s06-eventtrigger> ls
s06-eventtrigger>
```

Jika sudah berpindah folder ketikan truffle unbox react

```
s06-eventtrigger> truffle unbox react
✓ Preparing to download box
✓ Downloading
✓ cleaning up temporary files
✓ Setting up box
s06-eventtrigger> ls
```

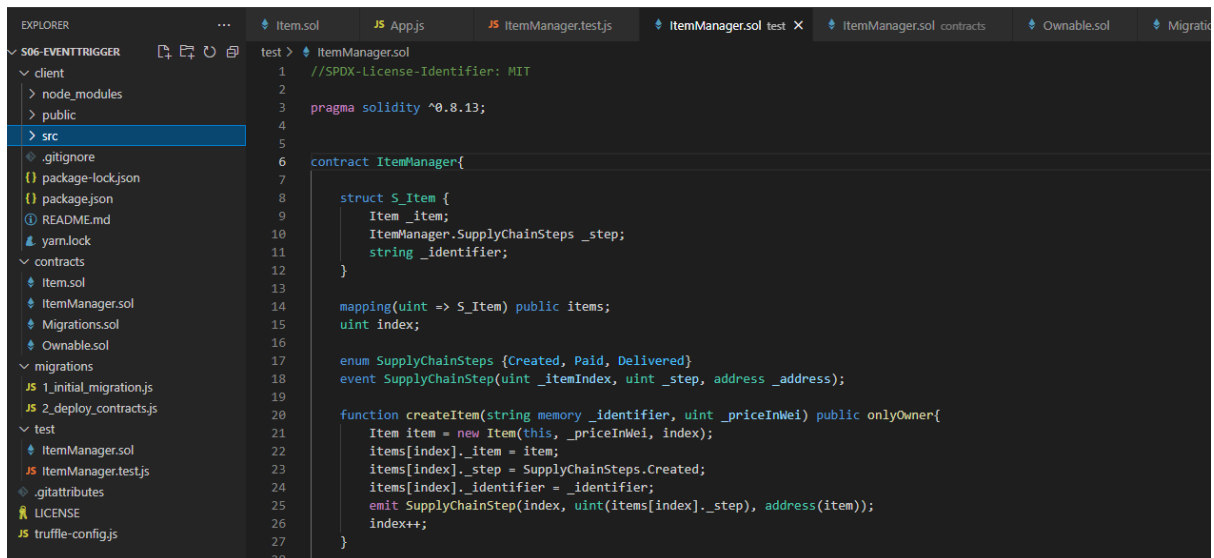
Lalu ls dan akan keluar seperti ini

```
PS C:\Users\Fajri\s06-eventtrigger> ls
```

```
Directory: C:\Users\Fajri\s06-eventtrigger

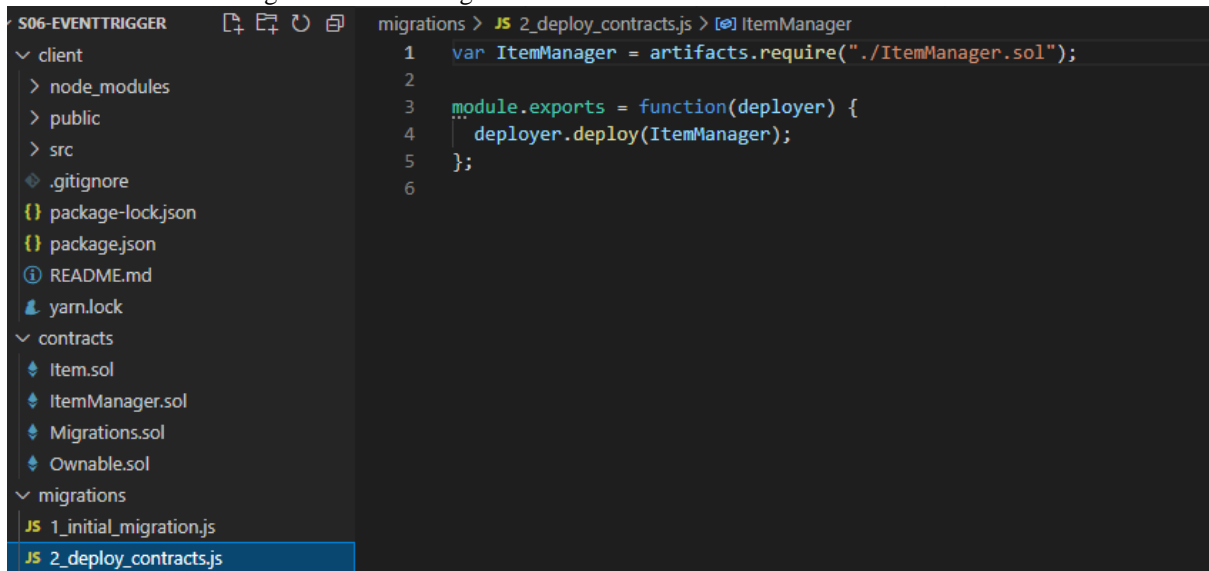
Mode                LastWriteTime         Length Name
----                -
d-----          4/21/2022   8:15 PM             client
d-----          4/21/2022   8:20 PM             contracts
d-----          4/21/2022   8:15 PM             migrations
d-----          4/21/2022   8:15 PM             test
-a-----          4/21/2022   8:15 PM              33 .gitattributes
-a-----          4/21/2022   8:15 PM             1075 LICENSE
-a-----          4/21/2022   8:28 PM             357 truffle-config.js
```

Jika sudah buka folder s06-eventtrigger tersebut dan cek dibagian contract jika ada file "Migrations.sol" silahkan hapus dan replace dengan file yang sudah dibuat sebelumnya dan akan menampilkan seperti ini



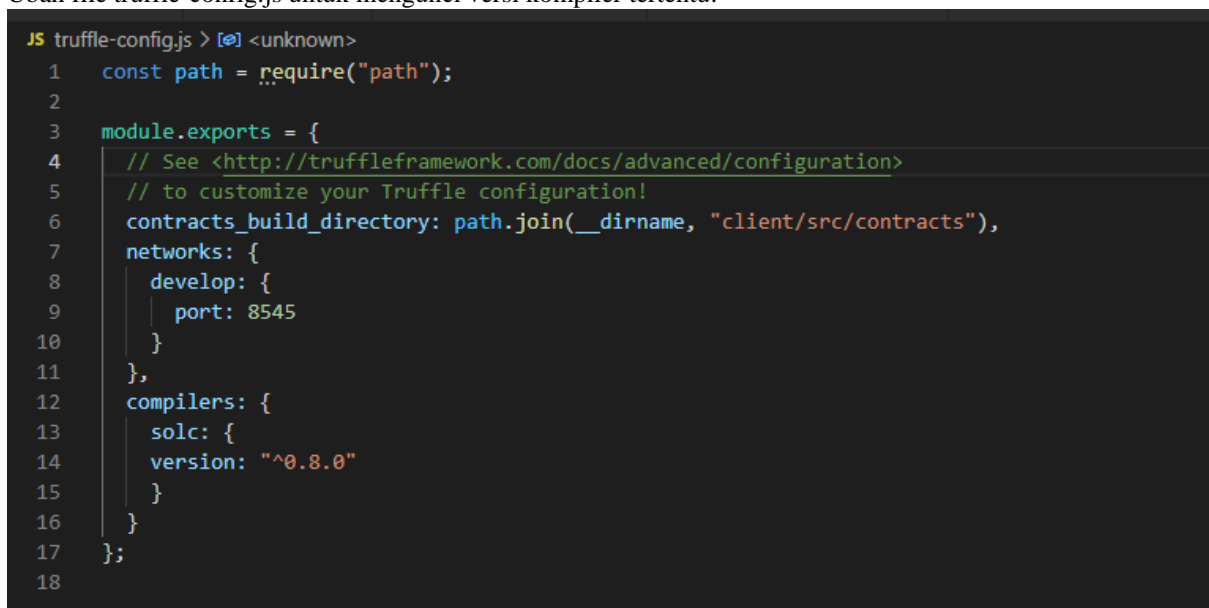
```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.13;
4
5
6 contract ItemManager{
7
8     struct S_Item {
9         Item _item;
10        ItemManager.SupplyChainSteps _step;
11        string _identifier;
12    }
13
14    mapping(uint => S_Item) public items;
15    uint index;
16
17    enum SupplyChainSteps {Created, Paid, Delivered}
18    event SupplyChainStep(uint _itemIndex, uint _step, address _address);
19
20    function createItem(string memory _identifier, uint _priceInWei) public onlyOwner{
21        Item item = new Item(this, _priceInWei, index);
22        items[index]._item = item;
23        items[index]._step = SupplyChainSteps.Created;
24        items[index]._identifier = _identifier;
25        emit SupplyChainStep(index, uint(items[index]._step), address(item));
26        index++;
27    }
28 }
```

Kemudian ubah file "migrasi" di folder migrasi/:



```
1 var ItemManager = artifacts.require("../ItemManager.sol");
2
3 module.exports = function(deployer) {
4     deployer.deploy(ItemManager);
5 };
6
```

Ubah file truffle-config.js untuk mengunci versi kompiler tertentu:



```
1 const path = require("path");
2
3 module.exports = {
4     // See <http://truffleframework.com/docs/advanced/configuration>
5     // to customize your Truffle configuration!
6     contracts_build_directory: path.join(__dirname, "client/src/contracts"),
7     networks: {
8         develop: {
9             port: 8545
10        }
11    },
12    compilers: {
13        solc: {
14            version: "^0.8.0"
15        }
16    }
17 };
18
```

```
truffle(develop)> migrate

Compiling your contracts...
=====
> Compiling .\contracts\Item.sol
> Compiling .\contracts\ItemManager.sol
> Compiling .\contracts\Ownable.sol
> Artifacts written to C:\Users\Fajri\s06-eventtrigger\client\src\contracts
> Compiled successfully using:
   - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang

Network up to date.
- Fetching solc version list from solc-bin. Attempt #1
- Fetching solc version list from solc-bin. Attempt #1
```

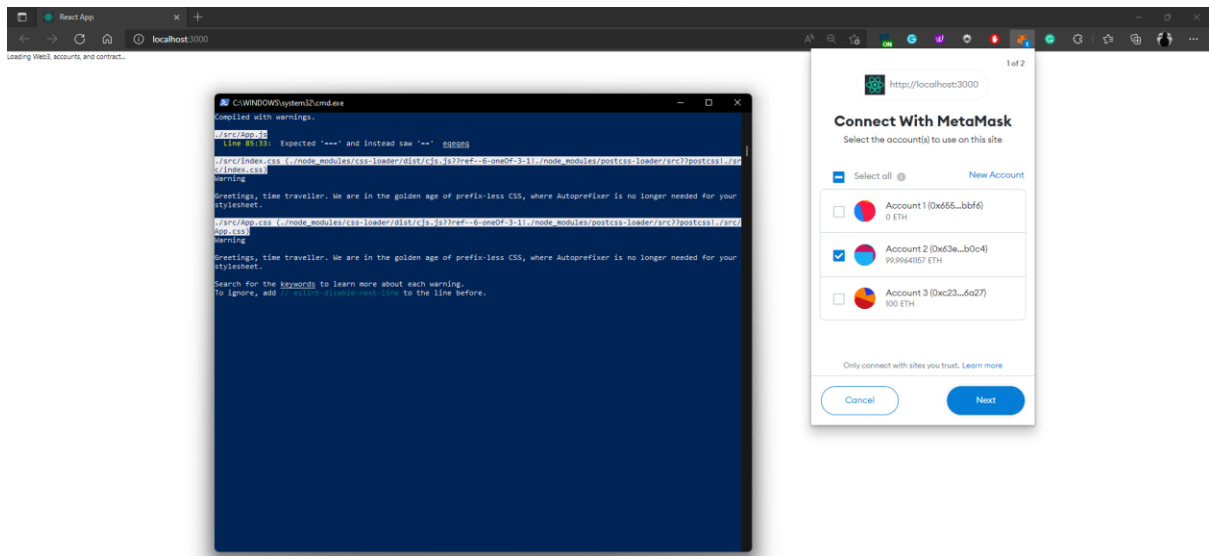
```
... Item.sol ... App.js ... ItemManager.test.js ... Ownable.sol ... ItemManager.sol test 1 ... Migrations.sol ... 2_deploy_contracts.js ... truffle-config.js ...

506- EVENT TRIGGER
  ↳ vitecode
  ↳ launch.json
  ↳ settings.json
  ↳ client
  ↳ node_modules
  ↳ public
  ↳ src
  ↳ contracts
  ↳ App.css
  ↳ App.js
  ↳ AppIndex.js
  ↳ getWeb3.js
  ↳ index.css
  ↳ index.js
  ↳ logo.svg
  ↳ serviceWorker.js
  ↳ .gitignore
  ↳ package-lock.json
  ↳ package.json
  ↳ README.md
  ↳ yarn.lock
  ↳ contracts
  ↳ Item.sol
  ↳ ItemManager.sol
  ↳ Migrations.sol
  ↳ Ownable.sol
  ↳ migrations
  ↳ 1_initial_migration.js
  ↳ 2_deploy_contracts.js
  ↳ test
  ↳ ItemManager.sol
  ↳ ItemManager.test.js
  ↳ glighttribes
  ↳ truffle-config.js

OUTLINE
TIMELINE
```

```
client > src > # App.js > App > listenPaymentEvent
1 import React, { Component } from 'react';
2 import ItemManager from '../contracts/ItemManager.json';
3 import Item from '../contracts/Item.json';
4 import getWeb3 from './getWeb3';
5
6 import './App.css';
7
8
9 class App extends Component {
10
11   state = { cost: 0, itemName: 'exampleItem1', loaded: false };
12
13   componentDidMount = async () => {
14     try {
15       // Get network provider and web3 instance.
16       this.web3 = await getWeb3();
17
18       // Use web3 to get the user's accounts.
19       this.accounts = await this.web3.eth.getAccounts();
20
21       // Get the contract instance.
22       const networkId = await this.web3.eth.net.getId();
23
24       this.itemManager = new this.web3.eth.Contract(ItemManager.abi, ItemManager.networks[networkId] && ItemManager.networks[networkId].address);
25
26       this.item = new this.web3.eth.Contract(Item.abi, Item.networks[networkId] && Item.networks[networkId].address);
27
28       this.setState({ loaded: true });
29     } catch (error) {
30       // Catch any errors for any of the above operations.
31       alert('Failed to load web3, accounts, or contract. Check console for details.');
```

Lalu buka baru power shell dan masuk ke folder client dan ketikan npm start



Sehingga akan muncul seperti ini dan akan menampilkan



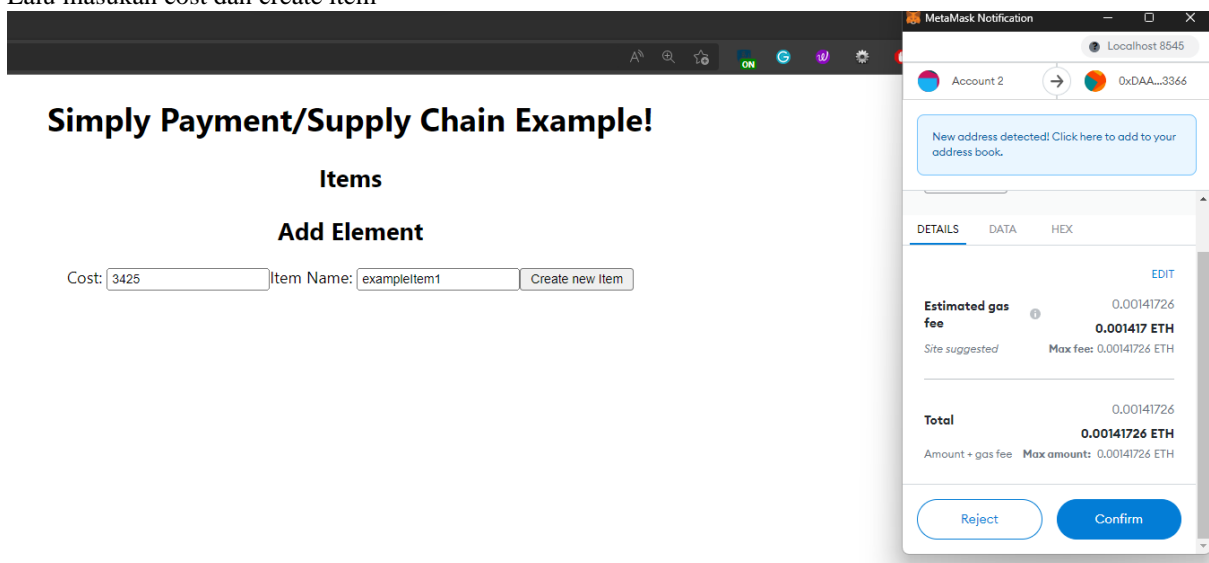
Simply Payment/Supply Chain Example!

Items

Add Element

Cost: Item Name:

Lalu masukan cost dan create item

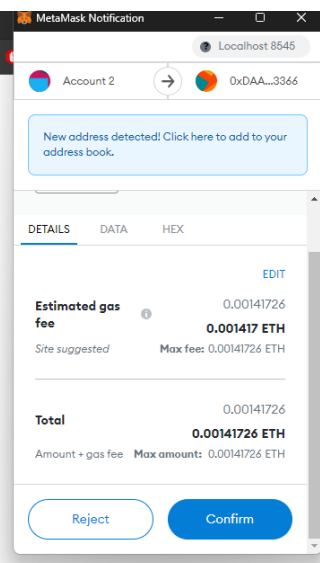


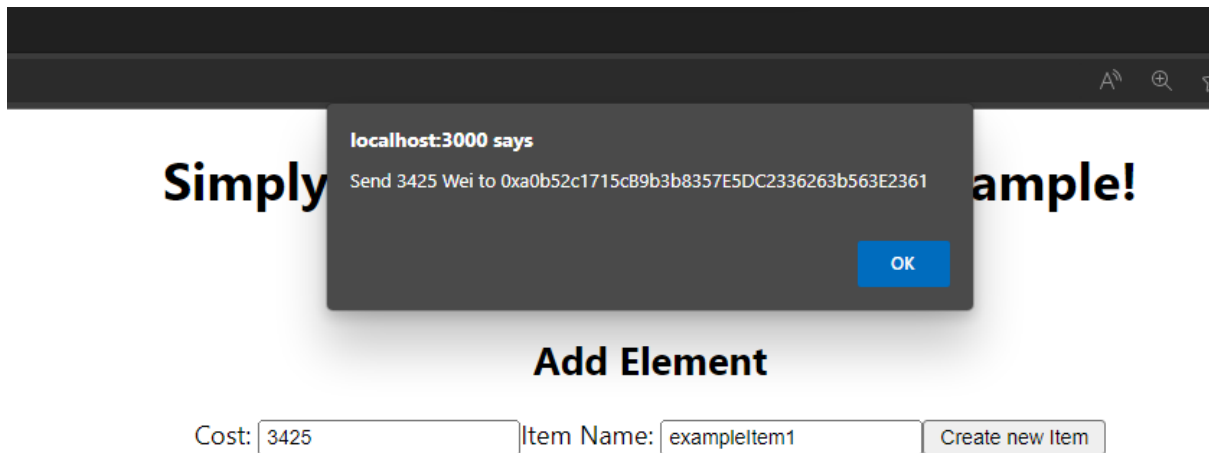
Simply Payment/Supply Chain Example!

Items

Add Element

Cost: Item Name:





Done semuanya selesai dan di truffle ada pengujian unit nah disini akan menerapkan unit test super sederhana dan lihat apakah bisa menguji item yang dibuat.

Pertama-tama, hapus tes di folder `"/test"`. Setelah it masukkan tes baru dengan code seperti berikut :

Jika sudah buka powershell baru dan tuliskan `truffle test` dan akan menampilkan seperti ini

```
PS C:\Users\Fajri\s06-eventtrigger\test> truffle test
Using network 'test'.

Compiling your contracts...
=====
✓ Fetching solc version list from solc-bin. Attempt #1
> Everything is up to date, there is nothing to compile.

Contract: ItemManager
  ✓ ... should let you create new Items. (726ms)

1 passing (976ms)
PS C:\Users\Fajri\s06-eventtrigger\test> █
```