LAB 3 Supply Chain Project

3.1 Project Supply Chain

3.1.1 Penggunaan Dunia Nyata Dari Project ini

Dapat menjadi bagian solusi dari supply chain

Pengiriman otomasi pada pembayaran

Pengumpulan payment tanpa orang tengah

3.1.2 Development Goal

Showcase Event-Triggers

Memahami fungsi tingkat rendah address.call.value()()

Pahami Alur Kerja dengan Truffle

Memahami Pengujian Unit dengan Truffle

Memahami Events dalam HTML

3.2 Smart Contract ItemManager

Pertama-tama kirta membutuhkan sebuah smartcontract yang Bernama “Management”, kita dapat menggunakan code dibawah ini

Text

Description automatically generated

Dengan ini menmungkinkan untuk menambahkan barang dan membayarnya, dengan bergeraknya maju didalam supply chain dan mentrigger pengiriman. Tetapi ada sebuah yang kita tidak sukai karena idealnya kita akan memberikan user dengan address yang mudah untuk pengiriman uang.

3.3 Smart Contract Item

Kita akan membuat satu lagi smart contract dengan nama “Item” untuk dari itu kit dapat menggunakan code dibawah ini

Text

Description automatically generated

Perlu kita ketahui pada solidity diatas 6.4 adanya penggantian “call.value(msg.value)(abi.encodeWithSignature("triggerPayment(uint256)", index))” menjadi “call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index))”

Lalu kita rubah juga pada smartcontract “Item Manager” kita agar dapat berinteraksi dengan smart contract “item” kita

Text

Description automatically generated

Dengan ini kita dapat hanya memberikan alamat ke customer dan mereka dapat langsung membayar dengan mengirim sekian Wei ke smartcontract. Tetapi smart contract kita masih belum aman. Kita membutuhkan sebuah fungsi kepemilikan.

3.4 Fungsi Kepemilikan

Normalnya kita akan menambahkan smartcontract OpenZeppellin dengan fungsi kepemilikannya. Tetapi Ketika penulisan ini, document belum di perbarui (masih di solidity 0.6.0). maka dari itu kita akan menambahkan fungsi kepemilikan oleh kita sendiri dan mirip dengan code OpenZeppllin. Kalian dapat membuat kode smart contract Bernama “Ownable.sol”

Text

Description automatically generated

Lalu kita rubah sedikit pada smartcontract “ItemManager” kita dan kita set untuk dapat di eksekusi oleh pemilik saja

Text

Description automatically generated

3.5 Install Truffle

Untuk instalasi truffle pada windows kita dapat menggunakan powershell, sedangkan yang berbasis UNIX bisa menggunakan terminal. Karena saya disini menggunakan windows, maka saya akan demonstrasikan dengan powershell.

Graphical user interface, text

Description automatically generated

Lalu buat folder disini saya menggunakan penamaan “s06-eventtrigger”

Text

Description automatically generated

Lalu unbox react boxnya

Text

Description automatically generated

Cek isi dari folder trufflenya

A picture containing text

Description automatically generated

Langkah selanjutnya kita buka text editor kita, disini saya memakai visual studio code. Setelah di buka arahkan ke folder yang kita sudah buat sebelumnya lalu di dalam folder contract hapus file “SimpleStorage.sol”

A screenshot of a computer

Description automatically generated with low confidence

Setelah mengghapus file “SimpleStorage.sol” kita masukan kontrak kita yang sebelumnya sudah kita siapkan dari remix. Jika ada pesan error maka kita abaikan saja terlebih dahulu.

A screenshot of a computer

Description automatically generated with low confidence

Setelah itu kita dapat merubah di evoirment kita agar dapat di compile

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Lalu dari sini kit dapat cek di console apakah ada yang masih belum tepat

Graphical user interface, text, application

Description automatically generated

3.6 Modifikasi HTML

Sekarang kita modifikasi HTML agar dapat ber interaksi dengan smartcontract kita di browser.

Buka di “client/app.js” dan modifikasi di bagian yang diberi warna kuning

A picture containing table

Description automatically generated

Lalu tambahkan render() seperti di bawah ini

Graphical user interface, text, application, email

Description automatically generated

Tambahkan dua fungsi, satu untuk handleInputChange, sehingga all input variable dapat di set dengan benar

Graphical user interface, text, application, email

Description automatically generated

Buka lagi terminal lalu jalankan npm dengan ini akan menjalankan server dengan port 300 di browser

Graphical user interface, application

Description automatically generated

3.7 Koneksikan dengan MetaMask

3.7.1 Apa Yang Kita Kerjakan?

Pada bagian ini kita akan mengkoneksikan react app dengan metamask dengan keystore ke sign transaction

3.7.2 Langkah-langkah

Pertama-tama koneksikan dengan metamask dengan network yang benar

A picture containing graphical user interface

Description automatically generated

Saat kami memigrasikan kontrak pintar dengan konsol Pengembang Truffle, maka akun pertama di konsol pengembang truffle adalah pemilik". Jadi, kami menonaktifkan MetaMask di Browser untuk berinteraksi dengan aplikasi atau kami menambahkan kunci pribadi dari truffle konsol pengembang ke MetaMask. Di Terminal/Powershell tempat Truffle Developer Console menjalankan gulir ke kunci pribadi di atas:

A picture containing text

Description automatically generated

Copy private key kita ke metamask

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Dari sini seharusnya akun kita akan muncul dengan saldo 100 Ether didalamnya

Sekarang kita tambahkan new item ke smartcontract

Graphical user interface, text, application

Description automatically generated

3.8 Listen to Payment

Sekarag kita tahu berapa yang kita harus bayar untuk spesifik alamat

Tambahkan fungsi dibawah ini ke app.js

Graphical user interface, text, application

Description automatically generated

Dan call fungsi ini Ketika kita inisialisasi aplikasi di “ComponenDidMount”

Graphical user interface, text, application, email

Description automatically generated

Ketika ada seseorang membayart maka item akan akan pop up dan meberitahukan anda untuk mengirim

Graphical user interface, text, application

Description automatically generated

3.9 Unit Test

Pengujian unit itu penting, itu tidak mungkin. Tetapi bagaimana cara menulis tes unit? Ada sesuatu yang istimewa di Truffle tentang pengujian unit. Masalahnya adalah di suite pengujian Anda mendapatkan abstraksi kontrak menggunakan truffle-contract, sementara di aplikasi normal Anda bekerja dengan instance web3-contract. Mari kita terapkan unit test super sederhana dan lihat apakah kita bisa menguji item yang dibuat. Pertama-tama, hapus tes di folder "/test". Mereka adalah untuk kontrak pintar penyimpanan paling sederhana yang tidak ada lagi. Kemudian tambahkan tes baru:

Test/ItemManager.test.js

Graphical user interface, text, application, email

Description automatically generated