Nama : M. Imam Whidyarto

NIM : 1103184089

Kelas : TK-42-PIL

LAB 3 : Supply Chain

ItemManager.sol

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.1;

import "./Allowance.sol";

contract SharedWallet is Allowance {

event MoneySent(address indexed \_beneficiary, uint \_amount);

event MoneyReceived(address indexed \_from, uint \_amount);

function withdrawMoney(address payable \_to, uint \_amount) public ownerOrAllowed(\_amount) {

require(\_amount <= address(this).balance, "Contract doesn't own enough money");

if(!isOwner()) {

reduceAllowance(msg.sender, \_amount);

}

emit MoneySent(\_to, \_amount);

\_to.transfer(\_amount);

}

function renounceOwnership() public override onlyOwner {

revert("can't renounceOwnership here"); //not possible with this smart contract

}

receive() external payable {

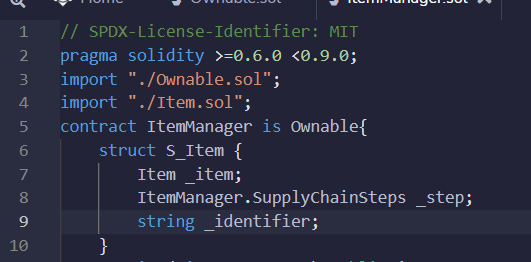
emit MoneyReceived(msg.sender, msg.value);

}

}

* Mendeploy smart contract
* Smart Contract ItemManager

Pertama kita membutuhkan Smart Contract ItemManager



Item.sol

// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.9.0;

import "./ItemManager.sol";

contract Item {

uint public priceInWei;

uint public paidWei;

uint public index;

ItemManager parentContract;

constructor(ItemManager \_parentContract, uint \_priceInWei, uint \_index) {

priceInWei = \_priceInWei;

index = \_index;

parentContract = \_parentContract;

}

receive() external payable {

require(msg.value == priceInWei, "We don't support partial payments");

require(paidWei == 0, "Item is already paid!");

paidWei += msg.value;

(bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));

require(success, "Delivery did not work");

}

fallback () external {

}

}

* Smart Contract Item

Kita akan membuat satu Smart Contract lagi yang bernama Item

Text

Description automatically generated

Owanable.sol

// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.9.0;

contract Ownable {

address public \_owner;

constructor () {

\_owner = msg.sender;

}

/\*\*

\* @dev Throws if called by any account other than the owner.

\*/

modifier onlyOwner() {

require(isOwner(), "Ownable: caller is not the owner");

\_;

}

/\*\*

\* @dev Returns true if the caller is the current owner.

\*/

function isOwner() public view returns (bool) {

return (msg.sender == \_owner);

}

}

* Fungsi kepemilikan

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

* Install Truffle

Untuk meninstall Truffle pada windows, kita menginstal berbasiskan CLI bisa menggunakan Windows Powershell dengan mengetikkan “ npm install -g [npm@8.7.0](mailto:npm@8.7.0) “

Text

Description automatically generated

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

Graphical user interface, application

Description automatically generated

Lalu unbox react boxnya

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”

Graphical user interface, text, application

Description automatically generated

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.

Graphical user interface, application

Description automatically generated

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

Text

Description automatically generated

Text

Description automatically generated