LAB 2 Shared Wallet

2.1 Project Shared Wallet

2.1.1   Kegunaan Dunia Nyata untuk Proyek ini

Tunjangan dana untuk anak per hari/minggu/bulan yang dapat di gunakan

Pegawai memberi tunjangan ke pewawai lainnya untu keperluan biaya perjalanan mereka

Bisnis memberikan kontraktor keperluan dana yang dapat di keluar untuk menggunakan anggaran

2.1.2   Pencapaian Dalam Pembuatan
Memiliki sebuah "on-chain wallet smart contract". Contract wallet dapat menyimpan saldo dan mengizinkan user untuk mengambil dana. Dapat memberikan tunjangan ke orang lain atau pada spesifik user berdasarkan alamat si user. Menggunakan Kembali smart contract yang telah di buat sebelumnya

2.2 Mendifinisikan Smart Contract
Dibawah ini adalah bentuk sederhana smartcontract. Ini dapat menerima ether dan memungkinkan untuk menarik Ether. Tetapi fungsi dari smart contract ini masih belum cukup berguna
Sharedwallet.sol

```solidity
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
import "./Allowance.sol";
contract SharedWallet is Allowance {
    function withdrawMoney(address payable _to, uint _amount) public {_to.transfer(_amount);
}

    receive() external payable {
        }
    }
```

2.3 Pemissions: Mengizinkan
Pada Langkah ini kita akan membatasi pengeluaran saldo ke pemilik wallet itu sendiri.
Bagaimana kita menetukan dari pemiliki?

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
contract SharedWallet {
address owner;
constructor() {
owner = msg.sender;
}
modifier onlyOwner() {
require(msg.sender == owner, "You are not allowed");
_;
}
function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
_to.transfer(_amount);
}
receive() external payable {
}
}
```

Pada code diatas kita juga dapat menambahkan fungsi "onlyOwner" untuk merubah ke fungsi "withdrawMoney"

2.4 Menggukan Kembali kontrak dari OpenZeppllin

Mempunyai logika "owner-logic" langsung didalam smart contract bukan lah hal yang mudah untuk di audit. Maka dari itu cobalah untuk memecah menjadi bagian-bagian kecil dan menggunakan smart contract yang telah di audit dari OpenZeppelin. Pada build OpenZepplin yang terbaru sudah tidak memiliki fungsi "isOwner" maka dari itu kita menambahkannya sendiri.

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
contract SharedWallet is Ownable {
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
_to.transfer(_amount);
}
receive() external payable {
}
}
```

2.5 Permission: Menambahkan Pengeluaran untuk Roles Luar

Pada Langkah ini kita menambahkan mapping, jadi kita dapat menyimpan address => uint amounts. Ini akan seperti array Ketika disimpan

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
contract SharedWallet is Ownable {
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
mapping(address => uint) public allowance;
function addAllowance(address _who, uint _amount) public onlyOwner {
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
require(_amount <= address(this).balance, "Contract doesn't own enough money");
_to.transfer(_amount);
}
receive() external payable {
}
}
```

Jika kita jeli pada smart contract yang kita buat ini masih ada bug, yaitu bug double spending

2.6 Improve/Fix Pengeluaran Guna Menghindari Double Spending

Dengan tidak mengurangi dana pada transaksi, seseorang adapat bertransaksi secara terus menerus dengan jumlah yang sama terus menerus. Pada kode dibawah ini kita mencoba membuat smart contract dengan mengurangi saldo untuk semuanya selain pemiliki.

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
contract Allowance is Ownable {
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
mapping(address => uint) public allowance;
function setAllowance(address _who, uint _amount) public onlyOwner {
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
allowance[_who] -= _amount;
}
}
contract SharedWallet is Allowance {
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);
```

2.7 Improve Struktur Smart Contract

Hingga disini kita sudah mengetahui fungsi basic dari structure smart contract. Untuk dapat mudah dibaca oleh developer lainnya, ada baiknya kita memecah dari fungsi smart contractnya

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
contract SharedWallet is Ownable {
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
mapping(address => uint) public allowance;
function addAllowance(address _who, uint _amount) public onlyOwner {
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
require(_amount <= address(this).balance, "Contract doesn't own enough money");
_to.transfer(_amount);
}
receive() external payable {
}
```

2.8 Menambahkan Event di Dalam Allowance Smart Contract

Disini kita akan menambahkan contract tetapi biarkan kosong terlebih dahulu

```solidity
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
contract Allowance is Ownable {
event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmou
mapping(address => uint) public allowance;
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
function setAllowance(address _who, uint _amount) public onlyOwner {
emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
allowance[_who] -= _amount;
}
}
contract SharedWallet is Allowance {
//...
```

2.9 Menambahkan Event didalam kontrak shared wallet

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
contract Allowance is Ownable {
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
mapping(address => uint) public allowance;
function setAllowance(address _who, uint _amount) public onlyOwner {
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
allowance[_who] -= _amount;
}
}
contract SharedWallet is Allowance {
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);
```

2.10   Menambahkan Library SafeMath untuk Operasi Aritmatik
Berdasarkan dari source code SafeMath Library operasi aritmatik pada solidity dibungkus didalam overlow. Ini dapat menghasilkan bug, karena programmer biasanya mengasumsi terjadi error overflow, yang dimana adalah prilaku standard di dalam high level programming languages. SafeMath mengembalikan pemikiran ini dengan mengembalikan transaksi jika operasi terjadi overflow. Pada update solidity terbaru tipe variable interger sudah tidak bisa overflow lagi, ini berlaku dari solidity versi diatas 0.8.

Jika anda masih menggunakan solidity versi dibawah 0.8 maka tambahkan kode dibawah ini

```solidity
pragma solidity ^0.6.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
import "https://github.com/OpenZeppelin/openzeppelin-contracts/contracts/math/SafeMath.sol" ;
contract Allowance is Ownable {
using SafeMath for uint;
event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmo
mapping(address => uint) public allowance;
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
function setAllowance(address _who, uint _amount) public onlyOwner {
//...
}
modifier ownerOrAllowed(uint _amount) {
//...
}
function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who].sub(_amount));
allowance[_who] = allowance[_who].sub(_amount);
}
}
contract SharedWallet is Allowance {
//...
}
```

Tetapi karena kita menggunakan solidity versi 0.8 keatas maka ini tidak perlu ditambahkan pada kode smart contract kita

2.11   Menghapus dari fungsi "Renounce Ownership"
Langkah selanjutnya hilangkan fungsi untuk menghapus pemilik. Kita hanya menghentikan ini dengan pengembalian. Tambahkan fungsi berikut ke Shared Wallet:

```solidity
contract SharedWallet is Allowance {
//...
function renounceOwnership() public override onlyOwner {
revert("can't renounceOwnership here"); //not possible with this smart contract
}
//...
}
```

2.12   Memindahkan Smart Contract Menjadi File yang Terpisah
Pada tahapan ini kita akan memisahkan file dan menggunakan fungsi import

SharedWallet.sol

```solidity
//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);
}
}
```

Allowance.sol

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.so
contract Allowance is Ownable {
event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmo
mapping(address => uint) public allowance;
function isOwner() internal view returns(bool) {
return owner() == msg.sender;
}
function setAllowance(address _who, uint _amount) public onlyOwner {
emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
allowance[_who] = _amount;
}
modifier ownerOrAllowed(uint _amount) {
require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
_;
}
function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
allowance[_who] -= _amount;
}
}
```