**SOURCE CODE**

It consists of three parts.

1) Smart contract

2) HTML

3) CSS

**Smart contract** :

1)IERC20.sol

2)SafeMath.sol

3)Cryptocoin.sol

**IERC20.sol** :

pragma solidity ^0.4.11;

interface IERC20 {

function totalsupply() constant returns(uint256 totalSupply);

function balanceOf(address owner)constant returns (uint256 balance);

function transfer(address to, uint256 value) returns (bool success);

function transferFrom(address from, address to, uint256 value) returns (bool success);

function approve(address spender, uint256 value) returns (bool success);

function allowance(address owner, address spender) constant returns (uint256 remaining);

event Transfer(address indexed from, address indexed to, uint256 value);

event Approval(address indexed owner, address indexed spender, uint256 value);

}

**SafeMath.so**l :

pragma solidity ^0.4.11;

/\*\*

\* @title SafeMath

\* @dev Math operations with safety checks that throw on error

\*/

library SafeMath {

function mul(uint256 a, uint256 b) internal returns (uint256) {

uint256 c = a \* b;

assert(a == 0 || c / a == b);

return c;

}

function div(uint256 a, uint256 b) internal returns (uint256) {

// assert(b > 0); // Solidity automatically throws when dividing by 0

uint256 c = a / b;

// assert(a == b \* c + a % b); // There is no case in which this doesn't hold

return c;

}

function sub(uint256 a, uint256 b) internal returns (uint256) {

assert(b <= a);

return a - b;

}

function add(uint256 a, uint256 b) internal returns (uint256) {

uint256 c = a + b;

assert(c >= a);

return c;

}

}

**Cryptocoin.sol** :

pragma solidity ^0.4.11;

import "./IERC20.sol";

import "./SafeMath.sol";

contract Cryptocoin is IERC20 {

using SafeMath for uint256;

uint public totalSupply =0;

string public constant symbol="CC";

string public name="Cryptocoin";

uint8 public constant decimals= 18;

//1 ether = 500CC

uint256 public constant RATE= 500;

address public owner;

mapping (address => uint256) balances;

mapping (address => mapping (address => uint256)) allowed;

function () payable{

createTokens();

}

function Cryptocoin()

{

owner=msg.sender;

}

function createTokens()payable

{

require(msg.value>0);

uint256 tokens=msg.value.mul(RATE);

balances[msg.sender]=balances[msg.sender].add(tokens);

totalSupply=totalSupply.add(tokens);

owner.transfer(msg.value);

}

function totalsupply() constant returns(uint256 totalSupply){

return totalSupply;

}

function balanceOf(address owner)constant returns (uint256 balance) {

return balances[owner];

}

function transfer(address to, uint256 value) returns (bool success) {

require(balances[msg.sender] >= value && value>0);

balances[msg.sender] = balances[msg.sender].sub(value);

balances[to] = balances[to].add(value);

Transfer(msg.sender,to,value);

return true;

}

function transferFrom(address from, address to, uint256 value) public returns (bool success) {

require(

allowed[from][msg.sender] >= value

&& balances[from] >= value

&& value>0

);

balances[from] = balances[from].sub(value);

balances[to] = balances[to].add(value);

allowed[from][msg.sender]-=value;

Transfer(from,to,value);

return true;

}

function approve(address spender, uint256 value) public returns (bool success) {

allowed[msg.sender][spender] = value;

Approval(msg.sender,spender,value);

return true;

}

function allowance(address owner, address spender) public view returns (uint256 remaining) {

return allowed[owner][spender];

}

event Transfer(address indexed from,address indexed to,uint256 value);

event Approval(address indexed owner,address indexed spender,uint256 value);

}

**2) HTML** :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>CryptoWallet</title>

<link rel="stylesheet" type="text/css" href="main.css">

<script src="./node\_modules/web3/dist/web3.min.js"></script>

</head>

<body>

<div class="container">

<h1>Sharether</h1>

<h2> Wallet Info</h2>

<p>Account:</p>

<p id="account"></p>

<p>Eth balance :</p>

<p id="balance"></p>

<p>Token balance :</p>

<p id="tokens"></p>

<h3>Send Tokens</h3>

<input id="to">

<input id="amount">

<button type="button" onclick="updatedata()">Submit</button>

<h4>Transactions</h4>

<div id="rightThing">

<label for="blockno" class="col-lg-2 control-label">Block no: </label>

<p id="blockno"></p></div>

<label for="tx.hash" class="col-lg-2 control-label">Tx.Hash </label>

<p id="tx.hash"></p>

<label for="tx.index" class="col-lg-2 control-label">Tx.Index : </label>

<p id="tx.index"></p>

<label for="frominfo" class="col-lg-2 control-label">From Address : </label>

<p id="frominfo"></p>

<label for="toinfo" class="col-lg-2 control-label">To Address : </label>

<p id="toinfo"></p>

<label for="amountinfo" class="col-lg-2 control-label">Amount : </label>

<p id="amountinfo"></p>

</div>

</div>

<script>

if (typeof web3 !== 'undefined') {

web3 = new Web3(web3.currentProvider);

} else {

web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

}

         web3.eth.defaultAccount = web3.eth.accounts[0];

var CryptoContract = web3.eth.contract(Contract ABI code)

const Crypto = CryptoContract.at('Contract address', (err, ctr) => {

return ctr

});

console.log(Crypto);

this.getetherBalance();

this.gettokenBalance();

function getetherBalance(){

Crypto.\_eth.getBalance("default account",(err,bal)=>{

if(!err){

this.coinbase = Crypto.\_eth.coinbase

console.log(coinbase);

document.getElementById('account').innerText = coinbase;

this.balance=web3.fromWei(bal,'ether').toNumber()

document.getElementById('balance').innerText = balance + " ether";

console.log(web3.fromWei(bal,'ether').toNumber()+" ether")

}

console.log(err)

})}

function gettokenBalance(){

Crypto.balanceOf("default account",(err,tkns)=>{

if(!err){

this.tokens=web3.fromWei(tkns,'ether').toNumber()

document.getElementById('tokens').innerText = tokens+" tokens";

console.log(web3.fromWei(tkns,'ether').toNumber()+" tokens")

}

console.log(err)

})}

function updatedata()

{

var to,amount

to=document.getElementById("to").value;

amount=document.getElementById("amount").value;

if(!web3.isAddress(to))

{

alert("Invalid address!")

this.to=null

return

}

if(isNaN(amount) || amount>tokens)

{

alert("Invalid amount!")

this.amount=null

return

}

Crypto.transfer(to,web3.toWei(amount,'ether'),(err,res)=>

{

if(!err)

{

console.log(res)

alert("Transferred ether successfully!!")

Crypto.Transfer({},{fromBlock : 0,toBlock:'pending'},(err,res)=>{

console.log(res)

if(res.args.to === Crypto.\_eth.coinbase || res.args.from === Crypto

.\_eth.coinbase){

document.getElementById('frominfo').innerText = res.args.from;

document.getElementById('toinfo').innerText = res.args.to;

document.getElementById('blockno').innerText = res.blockNumber;

document.getElementById('tx.hash').innerText = res.transactionHash;

document.getElementById('tx.index').innerText = res.transactionIndex;

document.getElementById('amountinfo').innerText =web3.fromWei(res.args.value).toNumber();

this.gettokenBalance()

this.getetherBalance()

}

})

this.amount=this.to=null

return

}else

console.log(err)

})

}

</script>

</body>

</html>

**3) CSS** :

body{

background-color: aliceblue;

padding: 2em;

font-family: 'Raleway','source sans pro','Arial';

}

.container{

width: 100%;

margin: 0 auto;

}

.rightThing{

width: 100%;

margin: 0 auto;

}

.leftThing{

width: 100%;

margin: 0 auto;

}

label{

display: block;

margin-bottom: 10px;

}

input{

padding: 10px;

width: 50%;

margin-bottom: 1em;

}

button{

margin: 2em 0;

padding: 1em 4em;

display: block;

}

#account{

padding: 1em;

background-color: white;

margin: 1em 0;

}

#balance{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#tokens{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#frominfo{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#toinfo{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#amountinfo{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#tx.hash{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#tx.index{

padding: 1em;

margin: 1em 0;

background-color:white;

}

#blockno{

padding: 1em;

margin: 1em 0;

background-color:white;

}