The bellow code is the multisignature wallet project using 5ire testnet

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract MultiSig {

    address[] public owners;

    uint public numConfirmationsRequired;

    struct Transaction{

        address to;

        uint value;

        bool executed;

    }

    mapping(uint=>mapping(address=>bool)) isConfirmed;

    Transaction[] public transactions;

    event TransactionSubmitted(uint transactionId,address sender,address receiver,uint amount);

    event TransactionConfirmed(uint transactionId);

    event TransactionExecuted(uint transactionId);

    constructor(address[] memory \_owners,uint \_numConfirmationsRequired){

        require(\_owners.length>1,"OwnersRequired Must Be Greater than 1");

        require(\_numConfirmationsRequired>0 && numConfirmationsRequired<=\_owners.length,"Num of confirmations are not in sync with the number of owners");

        for(uint i=0;i<\_owners.length;i++){

            require(\_owners[i]!=address(0),"Invalid Owner");

            owners.push(\_owners[i]);

        }

        numConfirmationsRequired=\_numConfirmationsRequired;

    }

    function submitTransaction(address \_to) public payable{

        require(\_to!=address(0),"Invalid Receiver's Address");

        require(msg.value>0,"Transfer Amount Must Be Greater Than 0");

        uint transactionId = transactions.length;

        transactions.push(Transaction({to:\_to,value:msg.value,executed:false}));

        emit TransactionSubmitted(transactionId,msg.sender,\_to,msg.value);

    }

    function confirmTransaction(uint \_transactionId) public{

        require(\_transactionId<transactions.length,"Invalid Transaction Id");

        require(!isConfirmed[\_transactionId][msg.sender],"Transaction Is Already Confirmed By The Owner");

        isConfirmed[\_transactionId][msg.sender]=true;

        emit TransactionConfirmed(\_transactionId);

       if(isTransactionConfirmed(\_transactionId)){

           executeTransaction(\_transactionId);

       }

    }

    function executeTransaction(uint \_transactionId) public payable{

       require(\_transactionId<transactions.length,"Invalid Transaction Id");

       require(!transactions[\_transactionId].executed,"Transaction is already executed");

        (bool success,) =transactions[\_transactionId].to.call{value: transactions[\_transactionId].value}("");

         require(success,"Transaction Execution Failed");

         transactions[\_transactionId].executed=true;

         emit TransactionExecuted(\_transactionId);

    }

    function isTransactionConfirmed(uint \_transactionId) internal view returns(bool){

         require(\_transactionId<transactions.length,"Invalid Transaction Id");

         uint confimationCount;//initially zero

         for(uint i=0;i<owners.length;i++){

             if(isConfirmed[\_transactionId][owners[i]]){

                 confimationCount++;

             }

         }

         return confimationCount>=numConfirmationsRequired;

    }

}