pragma solidity ^0.8.0;

contract TransactionManager {

    // Mapping of user balances

    mapping(address => uint256) public balances;

    // Mapping of user transaction history

    mapping(address => Transaction[]) public transactionHistory;

    // Struct to represent a transaction

    struct Transaction {

        uint256 amount;

        address sender;

        address recipient;

        uint256 timestamp;

    }

    // Event emitted when a transaction is made

    event TransactionMade(address indexed sender, address indexed recipient, uint256 amount);

    // Function to deposit funds

    function deposit() public payable {

        balances[msg.sender] += msg.value;

        transactionHistory[msg.sender].push(Transaction(msg.value, msg.sender, msg.sender, block.timestamp));

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

    }

    // Function to transfer funds

    function transfer(address \_recipient, uint256 \_amount) public {

        require(balances[msg.sender] >= \_amount, "Insufficient balance");

        balances[msg.sender] -= \_amount;

        balances[\_recipient] += \_amount;

        transactionHistory[msg.sender].push(Transaction(\_amount, msg.sender, \_recipient, block.timestamp));

        transactionHistory[\_recipient].push(Transaction(\_amount, msg.sender, \_recipient, block.timestamp));

        emit TransactionMade(msg.sender, \_recipient, \_amount);

    }

    // Function to get user balance

    function getBalance() public view returns (uint256) {

        return balances[msg.sender];

    }

    // Function to get user transaction history

    function getTransactionHistory() public view returns (Transaction[] memory) {

        return transactionHistory[msg.sender];

    }

}